On Edge Human Action Recognition Using Radar-Based Sensing and Deep Learning

Christian Gianoglio, *Member, IEEE*, Ammar Mohanna, Ali Rizik, Laurence Moroney, and Maurizio Valle, *Senior Member, IEEE*

***Abstract*—In this article, we propose a radar-based human action recognition system, capable of recognizing actions in real time. Range-Doppler maps extracted from a low-cost frequency-modulated continuous wave (FMCW) radar are fed into a deep neural network. The system is deployed on an edge device. The results show that the system can recognize five human actions with an accuracy of 93.2% and an inference time of 2.95 s. Raising an alarm when a harmful action happens is a crucial feature in an indoor safety application. Thus, the performance during the binary classification, i.e., fall vs nonfall actions, is also assessed, achieving an accuracy of 96.8% with a false-negative rate of 4%. To find the best tradeoff between accuracy and computational cost, the energy precision ratio of the system deployed on the edge is measured. The system achieves a 1.04 energy precision ratio value, where an ideal ratio would be close to zero.**

***Index Terms*—Action recognition, deep neural networks (DNNs), edge deployment, frequency-modulated continuous wave (FMCW) radar.**

I. Introduction

FALLS are a major public health concern and the main cause of accidental death in the senior population worldwide. Timely and accurate detection permits immediate assistance after a fall and, thereby, reduces complications of fall risk [1]. Edge-based approaches are essential to support time-dependent healthcare applications [2].

Due to the advantages of portability, low cost, and availability, wearable devices are regarded as one of the key types of sensors for fall detection and have been widely studied [3], [4], [5], [6]. The main drawback of these systems is the battery life which can limit the usability of the wearable devices. The second drawback is that the monitored subjects must always wear them, causing obvious discomfort, especially for the elderly.

Vision-based fall detection is another prominent method. Extensive effort in this direction has been demonstrated, showing promising results [7], [8], [9]. Although cameras are not as portable as wearable devices, they offer other advantages that deem them as decent options depending on the scenario: most static RGB cameras are not intrusive and wired, hence there is no need to worry about battery limitations. One major drawback of vision-based detection is the potential violation of privacy due to the levels of detail that cameras can capture, such as personal information, appearance, and visuals of the living environment. The second drawback is the high sensibility to clutters and environmental conditions (e.g., smoke, light, etc.). In addition, a vision-based approach could introduce issues related to color bias [10].

Ambient sensors (e.g., ultrasonic, WiFi antennas, radars, etc.) provide another nonintrusive means of fall detection. Ambient sensing is drawing more attention which can be attributed to being device-free for users and can solve the problem of people's privacy and color bias. Ultrasonic sensor network systems are one of the solutions for fall detection [11]. One drawback is that the waves are affected by environmental factors, such as humidity and smoke. These would affect the accuracy of the measurement. In [12], a fall detection approach uses WiFi signals, showing promising results in detecting falls. This approach is susceptible to security threats and has a limited range of efficiency.

Radars have become popular in recent years, proving to be an effective sensor to recognize human actions in cluttered environments [13]. In [14] and [15], the human action recognition is performed by a machine learning (ML) algorithm trained on hand-crafted features extracted from the radar signals. The main drawback is the time and effort required for the processing of the data and the feature extraction operation. Some other works propose deep neural networks (DNNs) to automatically extract features for human action recognition and fall detection. In [16], stack autoencoders (AEs) are used to automatically extract the features from the gray-scale spectrogram and to classify four activities including the fall. In [17], the authors combine convolutional neural networks (CNNs) and AEs to classify 12 actions based on micro-Doppler signatures. In [18], two DNNs

Manuscript received 3 August 2022; revised 6 March 2023; accepted 28 August 2023. Paper no. TII-22-3362. The paper has been submitted on 3rd August 2022 and accepted on 28th August 2023. *(Corresponding author: Ammar Mohanna.)*

Christian Gianoglio and Maurizio Valle are with the Electrical, Electronics, and Telecommunication Engineering and Naval Architecture Department (DITEN), University of Genoa, 16145 Genoa, Italy (e-mail: christian.gianoglio@unige.it; maurizio.valle@unige.it).

Ammar Mohanna is with the Assentify, Beirut 1499, Lebanon (e-mail: mohannaammar@gmail.com).

Ali Rizik is with the Department of Environmental, Land and Infrastructure Engineering (DIATI), University of Turin, 10129 Turin, Italy (e-mail: ali.rizik@edu.unige.it).

Laurence Moroney is with Google, Parkway Mountain View, CA 94043 USA (e-mail: lmoroney@google.com).

All the codes and data used in the work are available on GitHub: https://github.com/cosmiclabunige/ActionRecognition.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2023.3316164.

Digital Object Identifier 10.1109/TII.2023.3316164

automatically extract the features from the time series corresponding to the fast time of an ultra-wideband (UWB) radar return signal and classify fall actions. In [19], bidirectional long short-term memory (LSTM) networks classify activities, in real-time, based on the fusion of data collected using radar and wearable devices. In [20], LSTMs with and without bidirectional neurons classify the activities based on the micro-Doppler spectrograms. The data are considered as a continuous temporal sequence. In [21], the authors adopt a generative adversarial network (GAN) to enrich with synthetic samples a dataset containing a low number of micro-Doppler signatures representing human actions. Using this method, the capability of generalizing on new data is increased. In [22], a DNN takes binary-masked spectrograms as input. Those are computed on the signals of the UWB radar, classifying falls. In [23], the authors propose an algorithm to extract the optimal range bin from the range-Doppler spectrograms of a moving target for the subsequent time-frequency analysis, then a DNN built upon a pretrained model classifies the falls based on the optimal resulted spectrograms.

This article proposes a system for human action recognition based on deep learning (DL). The blocks of the system are designed for edge deployment. The data are collected using a low-cost frequency-modulated continuous wave (FMCW) radar connected to the edge device. The device transforms the signals into range-Doppler maps, treated as a series of images by the DNN. The performance of the system is evaluated both in terms of generalization accuracy and computational cost measured on the edge device. This evaluation covered the multiclass classification (i.e., five human action classes) and binary classification (i.e., fall versus nonfall classes).

The following are the novel contributions of this article.

1) The system is low-cost and deployed on the edge.
2) The proposed DNN classifies a series of range-Doppler maps through the medium of a time-distributed layer (TDL).
3) The DNN size achieves real-time edge inference.

The rest of this article is organized as follows. In Section II, the multiclass human action recognition method is described. In Section III, the radar specifications, data collection, training procedure, and edge deployment procedure are detailed. In Section IV, experimental results including both model and edge system assessment are presented. Finally, Section V concludes this article.

II. Methodology

In this section, the authors present the multiclass action recognition system. Fig. 1 presents the block diagram of the action elaboration, from the acquisition of the signals to the classification. The block diagram consists of four stages after the radar. In the following, all the stages are detailed.

A. 2-D FFT

In the 2-D FFT stage, the signals received by the Position2Go FMCW radar [24] are processed by the edge device (i.e., the Raspberry Pi 4) to obtain range-Doppler matrixes, also known as range-Doppler maps [25]. The radar receiver antenna

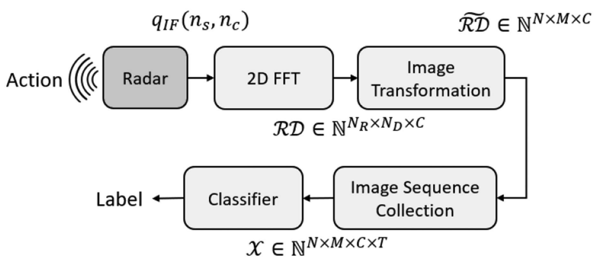

$q_{IF}(n_s, n_c)$
$\widetilde{\mathcal{RD}} \in \mathbb{N}^{N \times M \times C}$
Action
Radar
2D FFT
Image Transformation
$\mathcal{RD} \in \mathbb{N}^{N_R \times N_D \times C}$
Label
Classifier
Image Sequence Collection
$\mathcal{X} \in \mathbb{N}^{N \times M \times C \times T}$

Fig. 1. Block diagram of the action elaboration.

receives a delayed and attenuated copy of the transmitted wave. An I/Q mixer demodulates the received wave and returns an intermediate frequency (IF) signal. The signal is sampled by an analog-to-digital converter (ADC) and data are organized in a 2-D matrix. Following the notation adopted in [26], the matrix can be represented as

$$q_{IF}(n_s, n_c) = A_{IF} e^{j2\pi(f_b T_s n_s - f_D n_c T_{\text{PRI}})} \text{ with}$$
$$n_s = 0, \ldots, N_s - 1;\ n_c = 0, \ldots, N_c - 1 \quad (1)$$

where A_{IF} is proportional to received echo, T_s is the sampling period, f_D is the Doppler frequency shift, f_b is the beat frequency, T_{PRI} is the pulse repetition interval, and N_c is the number of chirps with N_s samples per chirp. According to (1), the signals in q_{IF} are filtered producing a filtered matrix $u(n_s, n_c)$. The range-Doppler (RD) map is computed as

$$RD^{(K_{sf})}(n_b, n_D) = \mathcal{F}_D^{N_D}\left\{\mathcal{F}_r^{N_R}\left\{u \times \omega_r^{(K_{sf})}\right\} \times \omega_D^{(K_{sf})}\right\}(n_b, n_D) \quad (2)$$

where u is the filtered matrix after removing the clutters [25], $\omega_r^{K_{sf}}$ and $\omega_D^{K_{sf}}$ are the Kaiser windows to be applied on the beat frequency and Doppler dimensions with a shape factor K_{sf}, and $\mathcal{F}_r^{N_R}$ and $\mathcal{F}_D^{N_D}$ are the range-FFT and Doppler-FFT outputting sequences of length N_R and N_D, respectively. After the processing, the RD map has the dimension $N_R \times N_D$. In this work, $N_R = N_D = 256$.

Since an RD map is a 3-D tensor, it can be considered an image and can be formalized as $\mathcal{RD} \in \mathbb{N}^{N_R \times N_D \times C}$, where N_R and N_D represent the height and width of the image, while C represents the number of channels (e.g., $C = 1$ in a gray-scale image and $C = 3$ in RGB format).

B. Image Transformation

A transformation can be applied to an image to convert it from one domain to another. Viewing an image in different domains enables the identification of features that may not be as easily detected in the initial domain. Among the image transformation techniques, edge detectors proved to increase accuracy in DL applications [27]. In this article, five transformations have been applied to the RD maps ($\mathcal{RD}$ in Fig. 1), three of which are based on edge detectors (i.e., Canny, Sobel, and Roberts). During the transformation, the RD maps are also resized to cope with the dimension of the input tensor of the classifier. Fig. 2 shows the different transformations applied on an exemplary RD map. The RD map format is RGB [Fig. 2(a)]. As aforesaid, this

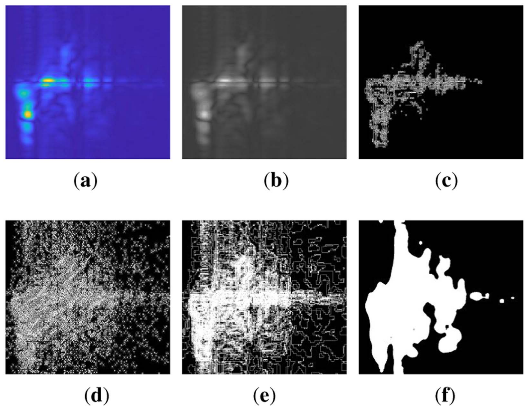


Fig. 2. Example of image transformations applied to the original image (a). (b) Grayscale. (c) Canny. (d) Roberts. (e) Sobel. (f) Binary.

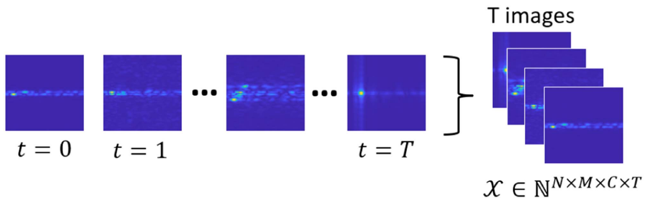


Fig. 3. Example of a 4-D tensor representing an action.

image is a 3-D tensor of dimension $N_R \times N_D \times C$. The applied transformations are the following.

1) *Gray* [Fig. 2(b)] decreases the number of channels C to 1, reducing also the color dependency.
2) *Canny* [Fig. 2(c)] uses a multistage algorithm to detect a wide range of edges in images.
3) *Sobel* [Fig. 2(d)] convolves the image with a small, separable, and integer-valued filter in the horizontal and vertical directions.
4) *Roberts* [Fig. 2(e)] approximates the gradient of an image through discrete differentiation which is achieved by computing the sum of the squares of the differences between diagonally adjacent pixels.
5) *Binary* [Fig. 2(f)] image is obtained by first applying the k-means clustering algorithm to an RD map, obtaining a gray-scale image. The clusters of pixels are then passed through a median filter to remove the outliers and set the nonoutlier pixels to white.

A transformed and resized RD map can be formulated as $\widetilde{RD} \in \mathbb{N}^{N\times M\times C}$.

C. Image Sequence Collection

The content of RD maps varies over time. Therefore, the sequence of RD maps is time-dependent. To classify the action, $T \times \widetilde{RD}$ images of each action are collected as a 4-D tensor datum, as shown in Fig. 3. The 4-D tensor can be formalized as $\mathcal{X} \in \mathbb{N}^{N\times M\times C\times T}$.

The adoption of sequences of RD maps results in an increment of classification accuracy over the single image, as demonstrated for human action recognition in [28].

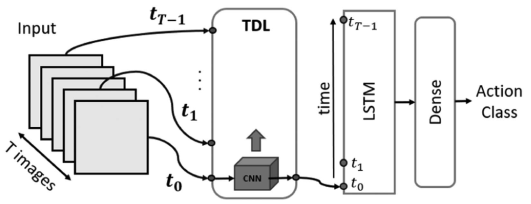


Fig. 4. Proposed DNN.

D. Classifier

Fig. 4 shows the proposed DNN (Classifier in Fig. 1) used to classify the 4-D tensor.

Human actions are dynamic and occur over time, and are thus captured in multiple RD maps, with each map constructed in approximately 30 ms, as described in Section III-A. To address the classification problem, we propose a hybrid model that utilizes a CNN to extract image features and an LSTM to capture the dependencies between consecutive maps. A TDL applies the same layer(s) to every time step of the input [29]. In this article, the TDL uses a CNN to extract T feature maps from the T images of the 4-D input tensor and the output of the TDL is a sequence of feature maps. An LSTM layer learns the dependencies between the sequence of feature maps. Finally, two dense layers are the output layers of the DNN. The first one is a fully connected network with an ReLU activation function. The second is made of Neu neurons, i.e., the number of classes, with a Softmax activation function to assign the action label.

Three CNNs have been designed. The use of CNNs aims to automatically extract features by learning the kernels (i.e., filters) that convolve with data. This practice is adopted to topple the domain-specific feature extraction, which is usually manually crafted by experts. The CNNs have been designed as a proof of concept for the feasibility of the action recognition system. The model's number of parameters is correlated with the size/performance ratio. Therefore, three CNN architectures of different sizes are adopted to investigate the effect of distinct number of layers on the performance of the overall system.

Fig. 5 shows the CNN reference architecture (called CNN2). The CNN2 comprises six main blocks: the first one, B1, consists of six 2-D convolutional layers taking as input a 3-D tensor of dimensions $N \times M \times C$, where N and M represent the dimension of the map and C is the number of channels. In this article, the range-Doppler maps [Fig. 2(a)] have $C = 3$, while all the transformed images [Fig. 2(b)–(f)] have $C = 1$. Each one of the five blocks (i.e., B2, B3, B4, B5, and B6) consists of a batch normalization layer followed by a 2-D average pooling and a dynamic number of 2-D convolutional layers (the number of convolutional layers decreases by one along with the blocks). The last block, B6, contains only one convolutional layer. Following this block, batch normalization and global average pooling layers are added. In particular, the global pooling flattens the last feature map.

Two different instances of the CNN2 architecture have been also taken into account. CNN1 has an architecture similar to CNN2 but with a lower number of layers: it contains five blocks instead of six, each block has a convolutional layer less than

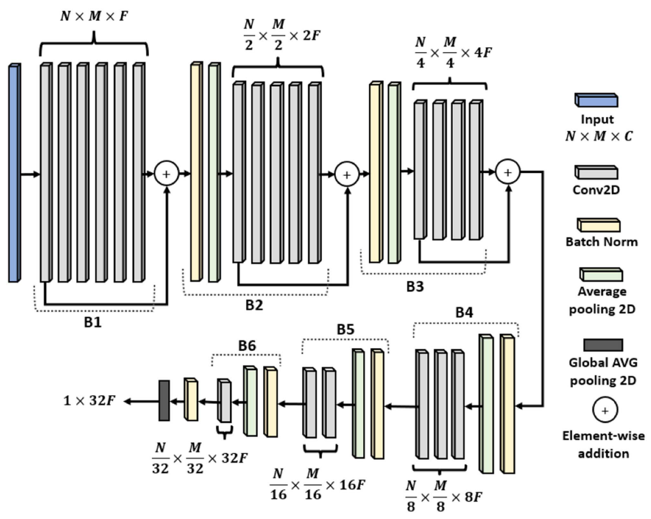


Fig. 5. CNN2 architecture.

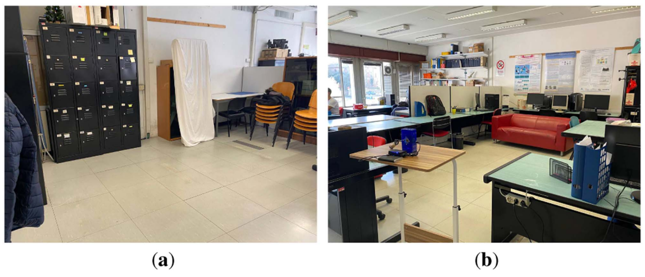


Fig. 6. Environments for data collection. (a) Environment 1 - Area ≈ 12 m^2. (b) Environment 2 - Area ≈ 23 m^2.

TABLE I
POSITION2GO RADAR SPECIFICATIONS

Parameters	Value	Parameters	Value
Sweep Bandwidth	200 MHz	Num Chirps/Frame	64
Center Frequency	24 GHz	Maximum Range	25 m
Up-Chirp Time	300 μs	Maximum Velocity	22.4 km/h
Steady-Chirp Time	100 μs	Range Resolution	0.9 m
Down-Chirp Time	100 μs	Velocity Resolution	0.4 km/h
Num Samples/Chirp	64	Sampling Rate	213.33 KHz

TABLE II
COLLECTED DATASET SUMMARY

Human Action	Classes Name	# of samples
Falling from a walking/ standing position	Fall	100
Falling from a laying position on a couch	Bed-Fall	100
Sitting down on a chair/couch	Sit	100
Standing up from a chair/couch	Stand	100
Walking	Walk	100

CNN2, and the global pooling is applied to the output of the fifth block (the last one in this case). On the opposite, CNN3 has one more convolutional layer in each block with respect to CNN2, thus presenting seven main blocks. In the following, DNN1, DNN2, and DNN3 will refer to the DNN presented in Fig. 4 that uses CNN1, CNN2, and CNN3, respectively.

III. EXPERIMENTAL SETUP

A. FMCW Radar Specifications

The radar used in the experiments is the Position2Go FMCW radar model operating on the 24-GHz ISM band, by Infineon Technologies [24]. The radar is equipped with a pair of arrays of microstrip patch antennas (one for transmitting and two for receiving) characterized by a 12-dbi gain and 19 × 76 degree beam widths, defining the field of view (FoV) in both elevation and azimuth axes, respectively. The sampling rate used for the data collection is 213 kHz to detect the high-frequency components of the signal (around 60 Hz) [28]. The development kit allows the user to implement and test several applications at the 24-GHz ISM bands, such as localizing, tracking, and collision avoidance [24]. Table I lists the radar parameters that are set for the data collection procedure. According to the specifications, each RD map is built upon 64 chirps. Every chirp consists of 300 μs of ramp-up, 100 μs of ramp-down, and 100 μs of steady-state period until the next chirp is generated. As a result, each map is thus computed over 32 ms of data. It takes 168 ms to transform a chirps frame into the RD map by applying the 2-D FTT. The next acquisition begins once the processing of the previous frame is complete, and the RD map has been saved.

B. Data Collection

The data have been collected in two different environments at the University of Genova, Italy. The environments contain clutters, such as desks, PCs, and metal lockers, as shown in Fig. 6.

A data sample consists of 15 consecutive range-Doppler maps acquired during 3 s of a human action performed in the FoV of the radar. The dataset contains five classes as follows.

1) *Fall*: The subject falls from a walking or standing state on a mattress.
2) *Bed-Fall*: A couch [Fig. 6(b)] represents a bed. It has been moved around the environment to perform the "fall from bed" action from different angles with respect to the radar's FoV.
3) *Sit*: The subject sits down on the couch or chairs positioned in different locations of the environment.
4) *Stand*: The subject stands up from the couch or chairs positioned in different locations of the environment.
5) *Walk*: The subject walks in the FoV of the radar.

The actions have been performed by five healthy subjects aged between 25 and 30. Only one subject performed one action at a time. To eliminate possible data-collection selection bias, the subject executed the same action in different ways (e.g., change the starting and/or ending positions with respect to the radar, perform the action faster/slower, etc.). In total, 100 data samples, i.e., 20 per subject, have been collected for each class (50 for each environment).

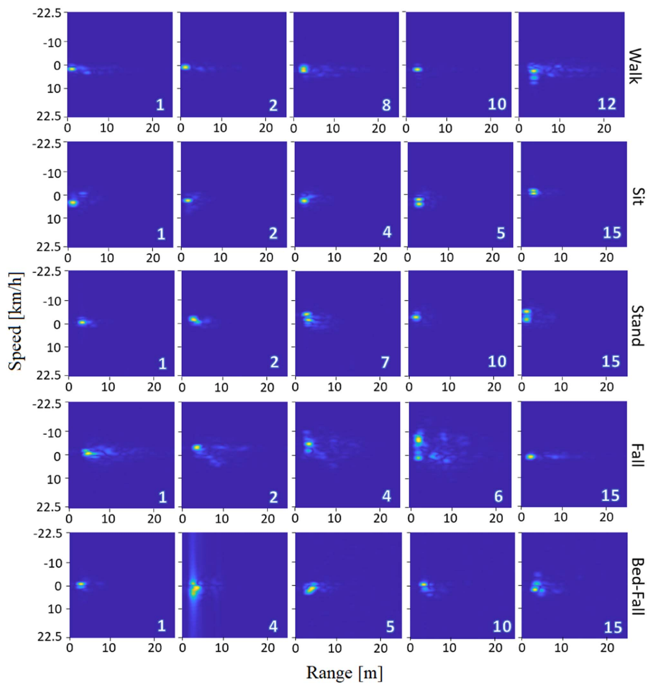


Fig. 7. Example of range-Doppler maps for the five actions.

Table II summarizes the collected data. The first column shows the performed actions, the second column the corresponding class to each action, and the last column reports the number of data acquired in each environment per class.

Five datasets have been generated from the $\mathcal{D}_{\text{Orig}}$ dataset by applying the transformation techniques described in Section II-A. The datasets can be formalized as

$$\mathcal{D}_d = \{(\mathcal{X}, y)_i\,; \mathcal{X}_i \in \mathbb{N}^{224\times224\times1\times15}; y_i \in \{\text{Fall}, \text{Bed-Fall}, \text{Sit}, \text{Stand}, \text{Walk}\}\,;\ i = 1, \ldots, 500\} \tag{3}$$

with $d \in \{\text{Gray}, \text{Canny}, \text{Sobel}, \text{Roberts}, \text{Binary}\}$.

Fig. 7 illustrates five RD maps per action, highlighting the most significant differences. Each map represents a snapshot of the action captured by the radar, where the x-axis denotes the distance of the human target from the radar, and the y-axis represents the speed retrieved by the Doppler FFT. A negative speed indicates that the moving target is approaching the radar, while a positive speed means that the target is walking away. The numbering in the bottom-right corner of each figure corresponds to the map position in the sequence. In the first row, the Walk action shows a target moving forward from the radar, with an increasing range and a speed lower than 5 km/h. The last map (number 12) displays a vertical distribution along the speed axis, indicating a Doppler spread caused by different ways in which the human body joints move, resulting in different Doppler frequencies. The Sit action in the second row shows a similar motion to the Walk action until the subject sits on the chair, resulting in a fixed position on the range axis in maps 5 and 15, and different frequencies due to body movements on the chair. On the other hand, the Stand action in the third row shows the subject sitting on a chair in maps 1 and 2, then standing up and starting to walk in maps 7, 10, and 15. The target approached the radar, resulting in a negative speed and a decreasing range. In the Fall action in the fourth row, the subject approaches the radar by walking in maps 1, 2, and 4. Map 6 captures the falling action, presenting a spread across both axes, most noticeable in the Doppler one. In the last map, the subject is lying on the mattress, resulting in a zero-Doppler frequency. The Bed-Fall action in the last row presents a similar pattern across all maps, with a zero Doppler frequency at a fixed range. Map 4 shows a frequency spread on the speed axis due to the falling action.

TABLE III
NUMBER OF PARAMETERS OF THE THREE DNNs

Model	Number of parameters
DNN1	304 117
DNN2	862 653
DNN3	2 962 677

C. DNNs Parameters

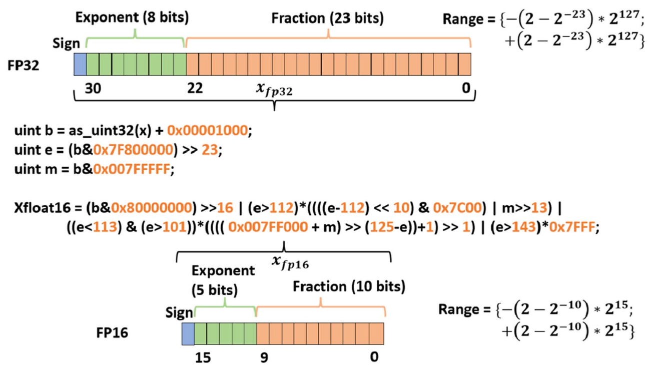
According to Fig. 5, the CNN input has dimensions $N \times M \times C$. In this article, we set $N = M = 224$ and $C = 3$ in case of an RGB image [i.e., Fig. 2(a)] and $C = 1$ for all the other transformations [i.e., Fig. 2(b)–(f)]. As mentioned in Section II-B, since the original RD maps (i.e., $\mathcal{RD}$ in Fig. 1) have dimension $256 \times 256 \times 3$, all the images have been resized to $224 \times 224 \times 3$ before applying the transformations. In the B1 block of all the models, the number of filters F is set to 8 while, in the further blocks, F is doubled while N and M are reduced by 50%. As a result, in the last block of CNN2 in Fig. 5 (i.e., $B6$ for CNN2) the output has dimension $7 \times 7 \times 256$. Consequently, the output of CNN1 has dimensions $14 \times 14 \times 128$, while the output of CNN3 has dimensions $3 \times 3 \times 512$. The LSTM layer that follows the TDL layer in Fig. 4 has 128 neurons for all the DNNs. The first dense layer has 64 neurons in all the DNNs. The output layer has five neurons, corresponding to the five classes. Table III presents the total number of parameters for each DNN. DNN1 refers to the architecture of Fig. 4 using CNN1. Equally, DNN2, and DNN3 use CNN2 and CNN3, respectively. Each of the three DNNs is trained using the six datasets (3).

D. Training

The training procedure has been implemented offline on a desktop PC with an Nvidia GeForce RTX 2080Ti GPU. All the DNNs are trained with the following parameters.

1) Adam optimizer with a learning rate $lr_{\text{start}} = 7e^{-5}$.
2) Number of epochs $ep = 200$.
3) Batch size $bs = 10$.
4) Loss function $lf = \textit{categorical cross entropy}$.
5) Early stop on validation accuracy with patience $p = 10$.

The stratified K-fold technique is adopted to provide fair results. According to the technique, a labeled dataset (population) is split into K parts containing the same proportion of data per class as in the population. This mechanism guarantees that the training and test sets contain the same proportion of data in each fold without affecting the approximation of the generalization accuracy. Each kth part is used, in turn, as the



```
uint b = as_uint32(x) + 0x00001000;
uint e = (b&0x7F800000) >> 23;
uint m = b&0x007FFFFF;

Xfloat16 = (b&0x80000000) >>16 | (e>112)*((((e-112) << 10) & 0x7C00) | m>>13) |
        ((e<113) & (e>101))*(((( 0x007FF000 + m) >> (125-e))+1) >> 1) | (e>143)*0x7FFF;
```

Fig. 8. Quantization from FP32 to FP16.

test set. The remaining $K-1$ folds compose the training set. In our experiments, K is set to 5. In this way, each fold contains 100 data samples, 20 per class. The generalization accuracy results presented in the next section are averaged over the five folds. Early stop criterion is adopted: from the $K-1$ folds used during the training, a validation set using a ratio of $1/4$ over the number of training data is extracted randomly. Eventually, a learning rate decay is adopted: during the training process, the learning rate is reduced every 10 epochs, multiplying it by 0.9. When the early stop criterion is satisfied (i.e., the validation accuracy decreases continuously for p epochs), the training procedure ends.

E. Edge Deployment

During inference, all stages (Fig. 1) are deployed on an edge device, where the tradeoff between generalization accuracy and the time for retrieving the action label becomes crucial. When it comes to classifiers, it is necessary to evaluate the trade-off between model size, latency, and accuracy. Hence, model optimization options must be considered during the model's conversion for edge deployment.

TensorFlow Lite (TF-Lite) is used to deploy DL models on mobile and edge devices, such as development boards and microcontrollers, offering optimization options for converting a TensorFlow model into the TF-Lite format. The adopted optimization option is quantization, which represents the model with lower precision [e.g., floating-point 16 (FP16) instead of the default floating-point 32 (FP32) representation]. This reduces the memory occupied by the model and the inference time. In this work, two quantization options have been used: no quantization, where the model parameters are represented as FP32, and FP16, where the parameters are converted to FP16, reducing the model size without affecting the inference time. The edge device used in this research is the Raspberry Pi4, which includes a high-performance 64-bit quad-core processor and 4 GB of RAM. The 4 GB version of the Raspberry Pi has been proven to be a reliable edge computing device for the signal processing of data collected by an FMCW radar [26], [30]. Posttraining quantization was performed on the host PC, where the TensorFlow-trained models were saved, converting all the weights of the network from FP32 to FP16. In the case of no quantization, the TensorFlow model was simply converted to TF-Lite format. Fig. 8 shows the representation of FP32 and FP16 numbers and provides a snippet of C++ code used to convert an FP32 number to an FP16 number according to the IEEE754 standard. Since the Raspberry Pi4 device does not support any operation but FP32, when the TF-Lite model is run, the FP16 numbers are cast back to FP32. Once the models are converted, they are deployed on the Raspberry device. With Python code, it is possible to load the sequence of RD maps, transform the original data (if necessary), and predict the label by running the TF-Lite model. In a real-time application, data are acquired from the radar by the Raspberry Pi4 using the script for data collection, and then the Python script is run to predict human action.

TABLE IV
AVERAGE ACCURACY ON THE 5-FOLDS

Dataset	Average Accuracy $\pm$ Standard Deviation		
	DNN1	DNN2	DNN3
$\mathcal{D}_{\text{Orig}}$	0.730 ± 0.044	0.920 ± 0.026	0.930 ± 0.028
$\mathcal{D}_{\text{Gray}}$	0.768 ± 0.065	$\mathbf{0.926 \pm 0.022}$	$\mathbf{0.932 \pm 0.029}$
$\mathcal{D}_{\text{Canny}}$	$\mathbf{0.828 \pm 0.044}$	0.892 ± 0.037	0.864 ± 0.022
$\mathcal{D}_{\text{Sobel}}$	0.740 ± 0.054	0.916 ± 0.038	0.884 ± 0.026
$\mathcal{D}_{\text{Roberts}}$	0.764 ± 0.071	0.880 ± 0.021	0.886 ± 0.037
$\mathcal{D}_{\text{Binary}}$	0.700 ± 0.072	0.860 ± 0.042	0.858 ± 0.017

The best accuracy for each DNN is emboldened.

IV. Results and Discussion

When introducing edge AI systems, it is crucial to assess two complementary aspects, i.e., the classification accuracy and the efficiency (e.g., inference time, power consumption, energy precision ratio, etc.).

A. Classification Accuracy

In this section, the results in terms of accuracy are presented. At first, the accuracy of the multiclass classification problem is assessed. Second, some performance metrics are evaluated in a binary classification problem. In this case, the classes Fall and Bed-Fall are considered fall actions (i.e., harmful), and the other classes are considered nonfall actions. The performance is computed on the multiclass classification results, by grouping the predicted labels into harmful and nonharmful classes.

Multiclass Classification

Table IV shows the average accuracy on the test set computed over the K folds for each of the three DNNs. The first column indicates the datasets and the other three columns report the average accuracy with their standard deviation. The accuracies, computed on the test sets, are averaged over the five folds. The best accuracy for each DNN is emboldened. Concerning DNN1, which contains the lowest number of parameters according to Table III, $\mathcal{D}_{\text{Canny}}$ presents the highest accuracy. For both DNN2 and DNN3, the best accuracies are achieved on the $\mathcal{D}_{\text{Gray}}$ dataset. The overall best accuracy (i.e., 93.2%), is obtained by DNN3 which contains the largest number of parameters (Table III), therefore the highest memory occupation and power consumption on an edge device.

Fig. 9 shows the confusion matrixes for the best DNNs, emboldened in Table IV, i.e., DNN1 trained with $\mathcal{D}_{\text{Canny}}$, DNN2 and DNN3 trained with $\mathcal{D}_{\text{Gray}}$. The predicted labels of the five

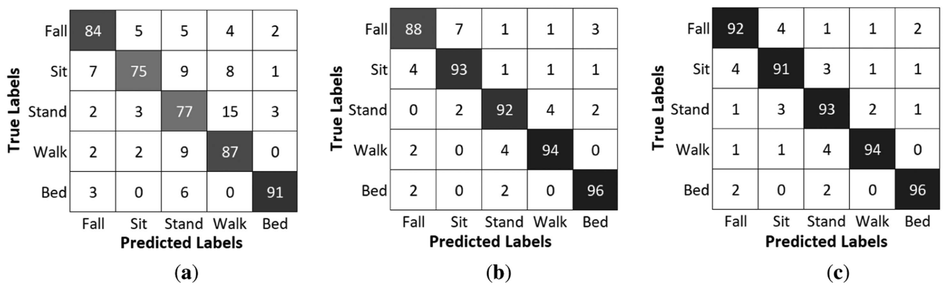


Fig. 9. Confusion matrixes of the three best models computed over the five folds. (a) DNN1 trained on $\mathcal{D}_{\text{Canny}}$. (b) DNN2 trained on $\mathcal{D}_{\text{Gray}}$. (c) DNN3 trained on $\mathcal{D}_{\text{Gray}}$.

TABLE V
METRICS COMPUTED ON BINARY CLASSIFICATION

Models	Acc	PR	SE	SP	FPR	FNR	F
DNN1 $\mathcal{D}_{\text{Canny}}$	0.930	0.923	0.900	0.950	0.050	0.100	0.911
DNN2 $\mathcal{D}_{\text{Gray}}$	0.964	0.955	0.945	0.970	0.030	0.055	0.950
DNN3 $\mathcal{D}_{\text{Gray}}$	0.968	0.960	0.960	0.973	0.027	0.040	0.960

folds have been merged, resulting in a hundred test samples for each class. The most reliable DNN for detecting falls, which coincides with the classes named Fall and Bed-Fall, as described in Section III-B, is DNN3. DNN2 shows a lower accuracy in detecting the Fall class. As one can notice from the first row in DNN3, the miss-classified samples are mostly classified as Sit. This is possibly due to the similarity between Fall and Sit actions.

Binary Classification

The following notation is used: True positives (TP) are fall actions correctly classified, false positives (FP) are nonfall actions incorrectly classified as Fall, true negatives (TN) are nonfall actions correctly classified, false negatives (FN) are fall actions incorrectly classified as NonFall. The following metrics are then adopted:

1) *Precision (PR)*, $\text{PR} = \frac{\text{TP}}{\text{TP+FP}}$, indicates how many predicted positive labels are positive.
2) *Recall or sensitivity (SE)*, $\text{SE} = \frac{\text{TP}}{\text{TP+FN}}$, indicates how much a model is accurate to predict the positive class.
3) *Specificity (SP)*, $\text{SP} = \frac{\text{TN}}{\text{TN+FP}}$, indicates how much a model is accurate to predict the negative class.
4) *False positive rate (FPR)*, $\text{FPR} = \frac{\text{FP}}{\text{FP+TN}} = 1 - \text{SP}$.
5) *False negative rate (FNR)*, $\text{FNR} = \frac{\text{FN}}{\text{FN+TP}} = 1 - \text{SE}$.
6) *F-score*, $\text{F} = 2 \times \frac{\text{PR*SE}}{\text{PR+SE}}$, balances between PR and SE.

The results presented in Table V prove that the proposed system is capable of distinguishing harmful from nonharmful actions. In particular, the FNR, which represents the percentage of harmful actions that do not activate the alarm since they are classified as nonharmful, is low, especially in DNN3. In general, DNN3 presents the best performance for all the metrics with respect to DNN1 and DNN2.

To further investigate the performance of the three best DNNs (in terms of accuracy) the receiver operating curves (ROC) and the area under curves (AUC) are computed on each fold. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR), varying the threshold for the score (i.e., the probability) based on which the output neurons assign the label. The AUC represents the area under the ROC curve and represents the degree of separability between classes. The AUC can be considered as an indicator of the performance of a classifier: the higher the value the higher the prediction accuracy. Fig. 10 reports the ROC and the AUC for the three best DNNs. Each plot presents seven lines: five colored lines refer to the ROC in each fold, the blue dotted line represents the average ROC curve, and the black dashed line corresponds to the baseline classifier, where FPR is equal to TPR. The plot legends report the AUC values corresponding to each fold and the average.

DNN3 presents the highest average AUC value. In general, DNN3 trained on $\mathcal{D}_{\text{Gray}}$ is a reliable model for detecting fall actions, achieving the highest accuracy in the multiclass classification problem, the lowest false negative rate in the binary classification problem, and the highest AUC value. Also, DNN2 could be a valuable option: despite it presenting slightly lower generalization performance with respect to DNN3, it contains less than one-third of the parameters than DNN3 (Table III). Thus, it is expected that, when deploying the models on the edge device, the computational cost of DNN2 is lower than the DNN3 one. In fact, during the deployment not only the generalization performance is important but also the computational cost. In this article, the computational cost is measured keeping into consideration all the stages of data elaboration (Fig. 1).

B. Edge System Assessment

The computational cost is evaluated as the inference time, power consumption, size of the model, and energy-precision ratio. Power consumption is estimated using a USB multimeter that is attached to the power supply of the edge device while running the inference. The energy-precision ratio (EPR) can be computed as $\text{EPR} = \text{Error} \times \text{EPI}$, where Error represents the classification error and EPI is the energy consumption per classified data item (Energy Per Item). According to Section III-E, two TF-Lite optimization methods are applied during the conversion of the three best DNNs in TF-Lite models. These classifiers and the previous stages are deployed on the Raspberry Pi4.

Table VI shows the computational cost and the classification accuracy of the quantized DNNs. All the results are averaged on the 500 test data used to evaluate the classification performance in the previous section. The first column depicts the models, the second column reports the quantization applied to each model for the deployment, and from the third to the last column, the

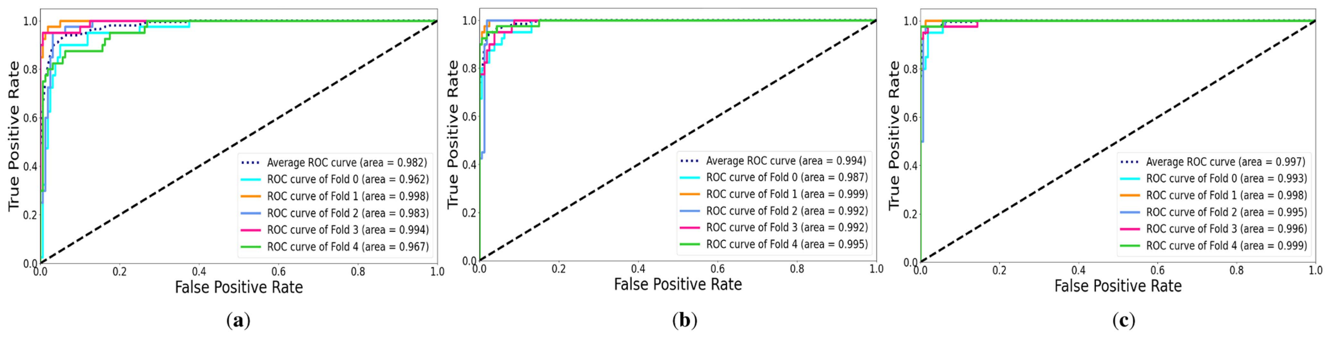


Fig. 10. ROC and AUC computed over the folds of the best performing models. (a) DNN1 trained on $\mathcal{D}_{\text{Canny}}$. (b) DNN2 trained on $\mathcal{D}_{\text{Gray}}$. (c) DNN3 trained on $\mathcal{D}_{\text{Gray}}$.

TABLE VI
SYSTEM ASSESSMENT ON RASPBERRY PI4

Models	Quant	Avg Acc	Time T (s)	Energy (J)	Size (MB)	EPR
DNN1	FP16	0.830	1.964	11.14	0.64	1.86
$\mathcal{D}_{\text{Canny}}$	FP32	0.828	1.967	11.40	1.2	1.96
DNN2	FP16	0.928	2.415	14.56	1.8	1.04
$\mathcal{D}_{\text{Gray}}$	FP32	0.926	2.415	14.80	3.5	1.09
DNN3	FP16	0.932	2.945	18.16	6.0	1.24
$\mathcal{D}_{\text{Gray}}$	FP32	0.932	2.948	18.26	11.9	1.23

table shows the five metrics averaged on 100 test samples over the five folds.

As expected, the first model (i.e., DNN1) has the lowest inference time and energy consumption, because of the lowest number of parameters that affect the model size. Straightforwardly, DNN3 presents the highest inference time and energy consumption. As can be noticed, the quantization to FP16 in DNN1 and DNN2 slightly improves the classification accuracy because it can act as a filter removing bits related to noise. The best tradeoff between generalization accuracy and inference time is achieved by DNN2, following EPR column of the table. The inference time of all the stages, with DNN2 as a classifier, is 2.4 s, thus guaranteeing the generation of an alarm in less than 6 s also considering 3 s for data acquisition.

C. Model Interpretability

Model interpretability is a crucial aspect in the analysis of DL models, especially in safety-critical applications where the prediction's correctness is fundamental. In this work, we employ GradCam++ to visualize the input regions that have the most significant impact on the network's decision. Figs. 11 and 12 show the results of GradCam++ [31] computed on the feature maps extracted from an average pooling layer in DNN3, specifically after the last residual block. GradCam++ produces heatmaps that highlight the parts of the input image that are most important in explaining the predicted class. In Fig. 11, the first row displays four RD maps of an input sequence consisting of 15 maps, where a fall action occurred correctly classified by the network. Maps 6 and 7 in the first row capture the falls, while map 1 refers to walking and map 10 to lying on the mattress after the fall. The second row shows the corresponding GradCam++ heatmap, highlighting the most relevant parts of the feature maps, which coincide with the most salient parts of the input. The third row depicts the pointwise multiplication of the GradCam++ heatmap with the input, clearly showing the regions of the input image that contributed the most to the classification. Similarly, in Fig. 12, the first row displays four RD maps of an input sequence consisting of 15 maps, where a fall action occurred, specifically maps 10 and 11. However, this sequence was misclassified as sitting by DNN3. The second and third rows show the corresponding GradCam++ heatmap and pointwise multiplication, respectively. The heatmaps look more spread, especially in map 11, with respect to the former example because of the higher level of noise that affected the measurements. Thus, even though GradCam++ seems to highlight salient regions of the input, probably the noise collected by the radar influences the eventual decision of the classifier.

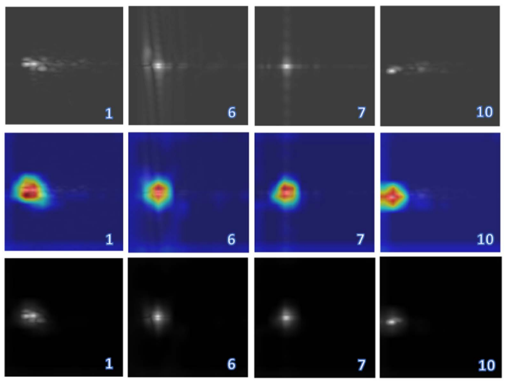


Fig. 11. Fall action correctly classified.

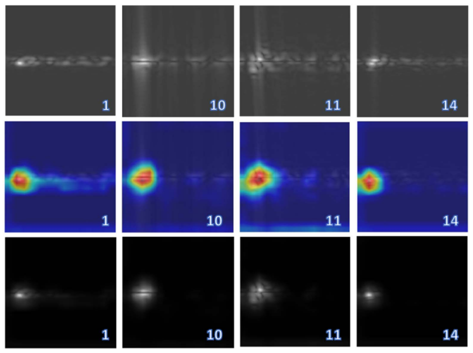


Fig. 12. Fall action classified as Sit.

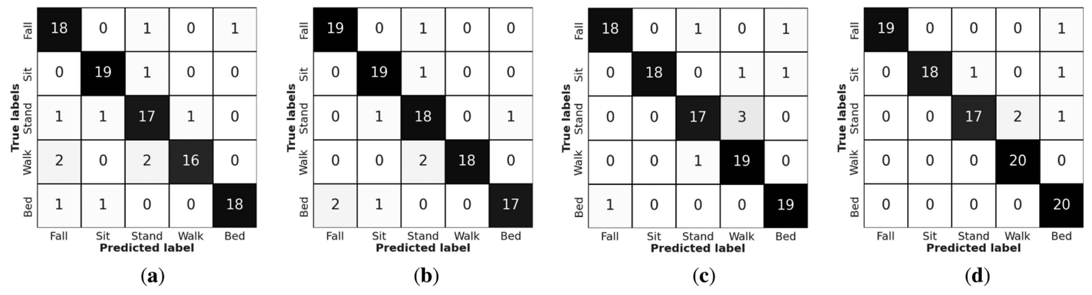


Fig. 13. Confusion matrixes of DNN2 and DNN3 computed on the new dataset with no transformation and in grayscale. (a) DNN2 tested on $\mathcal{T}_{\text{Orig}}$. (b) DNN2 tested on $\mathcal{T}_{\text{Gray}}$. (c) DNN3 tested on $\mathcal{T}_{\text{Orig}}$. (d) DNN3 tested on $\mathcal{T}_{\text{Gray}}$.

TABLE VII
COMPARISON WITH ADVANCED IMAGE RECOGNITION MODELS ON FIVE-CLASS CLASSIFICATION

Models	Num of Params	Trainable Params	Acc ± std
MNV2	2 987 973	729 989	0.790 ± 0.017
MNV2$_{FT}$	2 987 973	1 460 549	0.850 ± 0.028
Xcep	21 984 685	1 123 205	0.736 ± 0.032
Xcep$_{FT}$	21 984 685	4 286 853	0.846 ± 0.023
RN50V2	24 688 005	1 123 205	0.796 ± 0.024
RN50V2$_{FT}$	24 688 005	2 177 925	0.864 ± 0.043
DNN2	**862 653**	**861 645**	$\mathbf{0.920 \pm 0.026}$
DNN3	**2 962 677**	**2 960 645**	$\mathbf{0.930 \pm 0.028}$

The best accuracy for each DNN is emboldened. We emboldened the outcomes related to DNN2 and DNN3 for a better visualization.

TABLE VIII
COMPARISON OF METRICS WITH ADVANCED IMAGE RECOGNITION MODELS ON BINARY CLASSIFICATION

Models	Acc	PR	SE	SP	FPR	FNR	F
MNV2$_{FT}$	0.922	0.894	0.915	0.927	0.073	0.085	0.904
Xcep$_{FT}$	0.924	0.899	0.915	0.930	0.070	0.085	0.906
RN50V2$_{FT}$	0.932	0.917	0.920	0.940	0.060	0.080	0.915
DNN2	**0.944**	**0.925**	**0.940**	**0.947**	**0.053**	**0.060**	**0.931**
DNN3	**0.966**	**0.955**	**0.955**	**0.970**	**0.030**	**0.045**	**0.955**

The best accuracy for each DNN is emboldened. We emboldened the outcomes related to DNN2 and DNN3 for a better visualization.

D. Comparison With Advanced Image Recognition Models

Table VII presents a comparison of achieved results with state-of-the-art (SoA) algorithms for image recognition that utilize pretrained CNNs on the Imagenet dataset. Specifically, we considered MobileNetV2 (MNV2) [32], Xception (Xcep) [33], and ResNet50V2 (RN50V2) [34], all of which were encapsulated in the TDL layer (Fig. 4). We trained these deep architectures using two strategies: first, we adopted the transfer learning paradigm and only tuned the LSTM and dense layers while keeping the CNNs frozen; second, we unfroze the last layers of the CNNs and fine-tuned them together with the LSTM and dense layers. Table VII displays the total and trainable number of parameters, as well as the average accuracy on the five classes using the five-cross fold validation technique. The subscript "FT" denotes CNNs that were partially tuned by unfreezing the last layers. We used dataset $\mathcal{D}_{\text{Orig}}$ to train all networks since the RD maps were represented with three mandatory channels for these three pretrained CNNs. The results indicate that the DNN2 and DNN3 models achieve higher accuracy compared to all other networks, despite having a lower number of parameters. It is noteworthy that the partially fine-tuned models exhibit higher accuracy compared to their corresponding frozen models.

Table VIII presents a comparison of the metrics computed for the three deep networks based on image recognition models and the proposed approach. Only the partially fine-tuned models were included in the analysis, as they achieved better accuracy in the five-class classification problem. Once again, DNN2 and DNN3 exhibit superior performance across all the metrics.

Table IX shows the assessment of the three deep networks and the proposed models on the Raspberry Pi. We report only the results obtained with the FP16 quantization, as we have previously demonstrated that quantization has no significant impact on classification accuracy. The MobileNetV2-based network presents a lower inference time compared to our models, owing to the use of depthwise convolutions, leading to a lower energy consumption per predicted datum. However, our models offer a lower EPR as they achieve an accuracy improvement of over 6% compared to the MobileNetV2-based network.

TABLE IX
COMPARISON ON RASPBERRY PI4 WITH IMAGE RECOGNITION MODELS

Models	Quant	Avg Acc	Time T (s)	Energy (J)	Size (MB)	EPR
MNV2$_{FT}$	FP16	0.856	1.755	9.68	5.7	1.39
Xcep$_{FT}$	FP16	0.846	10.827	67.13	41.8	10.34
RN50$_{FT}$	FP16	0.862	8.389	51.93	47.0	7.18
DNN2	**FP16**	**0.920**	**2.074**	**11.76**	**1.7**	**0.94**
DNN3	**FP16**	**0.932**	**2.591**	**15.42**	**5.7**	**1.05**

The best accuracy for each DNN is emboldened. We emboldened the outcomes related to DNN2 and DNN3 for a better visualization.

TABLE X
GENERALIZATION PERFORMANCE IN ANOTHER ENVIRONMENT

Models	Acc2	PR	SE	SP	FPR	FNR	Acc5
DNN2 $\mathcal{T}_{\text{Orig}}$	0.960	0.950	0.950	0.967	0.033	0.050	0.900
DNN2 $\mathcal{T}_{\text{Orig}}$	0.970	0.975	0.950	0.983	0.017	0.050	0.910
DNN3 $\mathcal{T}_{\text{Gray}}$	0.980	0.975	0.975	0.983	0.017	0.025	0.920
DNN3 $\mathcal{T}_{\text{Gray}}$	0.980	0.950	1.00	0.967	0.033	0.000	0.940

E. Generalization Performance Assessment

A dataset comprising 100 samples (20 per class) was collected from a private house kitchen and bedroom, to evaluate the generalization performance of DNN2 and DNN3 on two

TABLE XI
COMPARING THE PROPOSED SYSTEM WITH RADAR-BASED HUMAN ACTION RECOGNITION AND FALL DETECTION IN SOA

Ref	Num Act.	Input	Acq Time	Model	Pros	Cons	Infer. Time
Jokanovic 2017 [16]	2	Spectrograms and range maps	4s	Two stacked autoencoders for data fusion of the two inputs, logistic regression classifier	High accuracy (97.1%)	No embedded implementation	-
Seyfioglu 2018 [17]	12	Spect.	4s	Convolutional autoencoder	High acc. (94.2%)	No embedded impl.	-
Sadreazami 2019 [18]	2	Raw data	15s	1-D CNN	High acc. (92.7%)	Classifier must wait 15s before decision, no embedded impl.	2.36s on Intel Core i7 cpu
Li 2020 [19]	6	Spect. bins and IMU data	Cont.	Bi-LSTM with sensor fusion for continuous monitoring	High acc. (96%) and continuous monitoring	No embedded impl.	-
Shrestha 2020 [20]	6	Spect. bins	Cont.	Bi-LSTM for a cont. monitoring	Quite a high acc. (90%) and cont. monitoring	No embedded impl.	-
Erol 2020 [21]	8	Spect.	4s	Adversarial deep network creating synthetic data to train a deep network for classification	High acc. (93%) on true data collected in different environments	No embedded impl.	-
Sandreazami 2021 [22]	2	Spect. + image transformation	15s	2-D CNN	High acc. (95%)	Very big network (>70M params), very long input, no embedded impl.	-
Zhu 2021 [35]	6	Spect.	3s	Tiny 1-D CNN based on pointwise and depthwise convolutions	High acc. (95%), embedded impl., tiny model, fast inference	No fall detection, authors did not manage the deployment of data preprocessing	~5ms on a smartphone
Stadelmayer 2021 [36]	6	Raw data	2s	1-D CNN with custom filters to extract features from time/frequency domain	High acc. (99.5%), quite a tiny model (130K params)	No fall detection, no embedded impl.	-
Lai 2021 [37]	7	Spect.	3s	1-D CNN with time/frequency attention mechanism	High acc. (98.1%), quite a tiny model (<200K params)	No fall detection, No embedded impl.	-
Zhu 2020 [38]	7	Spect.	3s	Hybrid model 1-D CNN + LSTM for extracting spatio-temporal characteristics from spect.	High acc. (98.3%), quite a tiny model (~200K params)	No fall detection, No embedded impl.	-
Qian 2023 [39]	8	Spect. computed with 3 different processing	4s	3 CNN to extract features from the three spect. feeding 3LSTM + 3D CNN to extract features and a fully connected to fuse all the feature maps	High acc. (94.75%)	No fall detection, No embedded impl., very big model (>70M params)	-
Ou 2022 [40]	2	2 channels raw data	2.4s	Tiny 1-D CNN based on depthwise convolutions	Quite a high acc. (90.1%), embedded impl., tiny model model, fast inference	Authors did not manage the deployment of data preprocessing	3.2ms on a smartphone
Seyfioglu 2019 [42]	7 and 11	Spect.	4s	Residual 2-D CNN, the dataset was enhanced with synthetic data built with simulations	High acc. (97% and 95.7%)	No embedded impl., quite a big network (>5M)	-
Chen 2019 [43]	7	Raw data	3s	1-D CNN with inception densely blocks	High acc. (96.1%), quite a tiny model (365K params)	No embedded impl.	-
Li 2022 [44]	6	Spect.	5s	Semisupervised approach based on deep learning with adversarial domain transfer	Good acc. (87.6%) with 10% labeled dataset	No embedded impl.	-
Ding 2019 [41]	7	Sequences of RD maps	Cont.	Cont. monitoring extracting features from each map. A set of features on multiple maps is used to classify an action by a majority of voting	High acc. in single motion recognition (95%) and in continuous motion (91.9%)	No embedded impl.	-
Proposal	**5**	**Sequences of RD maps**	**3s**	**Image transformations on inputs, CNN encapsulated in a TDL + LSTM**	**High acc. (92.6% and 93.2%), embedded impl.**	**The most accurate model is not so tiny (~3M param)**	**2.42s and 2.95s on a Raspberry**

different subjects aged 26 and 32 years old. Based on the previous results, the networks were tested on the dataset using no transformation and grayscale. The datasets can be represented as $\mathcal{T}_d = \{(\mathcal{X}, y)_i; \mathcal{X}_i \in \mathbb{N}^{224\times224\times1\times15}; y_i \in \{\text{Fall, Bed-Fall, Sit, Stand, Walk}\}; i = 1,\ldots,100\}$ with $d \in \{\text{Orig, Gray}\}$. Table X summarizes the performance of the classification of the two networks on $\mathcal{T}_d$. The first six columns show the metrics computed in the binary classification, while the last column presents the accuracy achieved in the five-class problem. The results indicate that both DNN2 and DNN3 have good generalization performance on $\mathcal{T}_d$ with both transformations, with DNN3 achieving higher accuracy in both the binary and five-class classification. In particular, DNN3 tested on $\mathcal{T}_{\text{Gray}}$ achieves a 0% of false negatives recognizing all the fall actions and having a low rate of false positives. Fig. 13 visualizes the confusion matrixes.

TABLE XII
COMPARING RADAR WITH WEARABLE AND CAMERA-BASED SYSTEMS

Wearable-based systems		
Ref	**Pros**	**Cons**
[3]–[6], [46], [47]	Low-cost devices. Accuracy is even higher than 90% in action recognition and fall detection. Simple algorithms achieve high accuracy and can be easily deployed on constrained devices. Simultaneously monitoring multiple people. They guarantee privacy.	Each monitored person must always wear a device risking damaging it and causing discomfort, especially for the elderly.
Cameras-based systems		
Ref	**Pros**	**Cons**
[7]–[9], [48]–[50]	High accuracy and they do not need any device to be worn by a person. With more complex processing and deep learning techniques, it is possible to monitor more persons in the same room	Privacy issues for the monitored people. Many of the devices are susceptible to changes in lighting conditions and environmental clutter. Sometimes it is necessary to install multiple devices to monitor a single environment. Classifying actions with these systems may require the use of complex techniques.
Radars-based systems		
Ref	**Pros**	**Cons**
Table XI refs	High accuracy and they do not need any device to be worn by a person. They can monitor cluttered environments. They guarantee privacy.	Some devices may not support the monitoring of multiple people, which may require the use of more expensive equipment or advanced processing techniques. Classifying actions with these systems may require the use of complex techniques.

F. Comparison With Radar-Based Works

Table XI presents a comparison of the proposed system with the most relevant SoA systems in radar-based human action classification using ML. The table provides detailed information on the number of actions classified, the type of input processed by each model, the acquisition time for each data point, the classification model, the strengths and weaknesses of each approach, and the inference time if available. The last row of the table presents the proposal and includes an analysis of the two most accurate models, DNN2 and DNN3, which were trained using $\mathcal{D}_{\text{Gray}}$. All of the works achieved high accuracy in radar-based human action recognition, but some did not specifically focus on fall detection, such as [35], [36], [37], [38], and [39]. Several studies have focused on binary classification for fall detection among common daily activities, such as walking, sitting, standing, and other nonfalling actions [16], [18], [22], [40]. Most of the works used raw data or computed spectrograms to train the classifiers, while only one [41] collected sequences of RD maps to continuously monitor human actions. Only two works addressed the implementation of classifiers on embedded devices, namely, [35] and [40]. Although the authors designed tiny models with a very low inference time on the edge, they did not deploy the data preprocessing stages on the edge which were performed offline, e.g., blocks 2-D FFT, image transformation, and image sequence collection in Fig. 1. To the best of our knowledge, this is the first work on radar-based human action recognition with fall detection in which the entire pipeline has been deployed on the edge.

G. Comparison With Wearable and Cameras-Based Systems

In the SoA, many works addressed human action recognition and fall detection with a multitude of sensors, such as wearables, cameras, and radars. A recent review can be found in [45]. Table XII summarizes most of the highly cited research on the topic. Each sensor category has its own strengths and weaknesses, which are highlighted in the table. Overall, the studies achieve high levels of accuracy, comparable to those obtained with radar-based approaches (see Table XI). The choice of which type of sensor to use depends on the specific application. Wearable-based systems are ideal when users can consistently wear the sensors, as they can monitor people at all times, are relatively low-cost, may require less complex processing, and guarantee privacy. Conversely, when wearable sensors are not an option, radars or cameras can be chosen instead. Radars guarantee privacy, can monitor people in cluttered environments, and are not affected by changes in lighting conditions. Camera-based systems are more accurate at detecting multiple people in the same room, but they do not guarantee privacy, and clutter can interfere with monitoring.

V. CONCLUSION

This article presents an on-the-edge radar-based human action recognition system using DL. The system uses sequences of range-Doppler maps extracted from a low-cost FMCW radar. A time-distributed layer processes the sequence of range-Doppler maps. The results showed that the model with the highest number of parameters (i.e., DNN3) achieves the best accuracy (93.2%) in the five-class classification using grayscale data transformation. Moreover, the same model distinguished harmful from nonharmful actions with an accuracy of 96.8% and a false-negative rate of 4%. Using a radar that has higher performance would certainly help reduce the classification error and the false negative rate. The proposed system was deployed on a Raspberry Pi4. The results showed that the system that uses DNN2 achieves the best tradeoff, with a slight drop in accuracy, i.e., lower than 1%, with respect to DNN3 and an inference time lower than 2.5 s.

In future works, we will tackle the multitarget classification problem by using radars that can overcome the limitations of range and speed resolution. To address this scenario, we will also explore techniques for separating targets and subsequently classifying their actions. In addition, we will focus on implementing a continuous monitoring system to reduce the acquisition and classification time, and we will test the system on a sample of the elderly. Finally, we will leverage semisupervised and unsupervised techniques with the aim of detecting a broader variety of human actions.

References

[1] M. Ronthal, "Gait disorders and falls in the elderly," *Med. Clin.*, vol. 103, no. 2, pp. 203–213, 2019.

[2] P. Pace, G. Aloi, R. Gravina, G. Caliciuri, G. Fortino, and A. Liotta, "An edge-based architecture to support efficient applications for healthcare industry 4.0," *IEEE Trans. Ind. Inform.*, vol. 15, no. 1, pp. 481–489, Jan. 2019.

[3] A. Shahzad and K. Kim, "FallDroid: An automated smart-phone-based fall detection system using multiple kernel learning," *IEEE Trans. Ind. Inform.*, vol. 15, no. 1, pp. 35–44, Jan. 2019.

[4] C. Wang et al., "Low-power fall detector using triaxial accelerometry and barometric pressure sensing," *IEEE Trans. Ind. Inform.*, vol. 12, no. 6, pp. 2302–2311, Dec. 2016.

[5] W. Saadeh, S. A. Butt, and M. A. B. Altaf, "A patient-specific single sensor IoT-based wearable fall prediction and detection system," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 27, no. 5, pp. 995–1003, May 2019.

[6] A. B. Mesanza et al., "Machine learning based fall detector with a sensorized tip," *IEEE Access*, vol. 9, pp. 164106–164117, 2021.

[7] M. Yu, A. Rhuma, S. M. Naqvi, L. Wang, and J. Chambers, "A posture recognition-based fall detection system for monitoring an elderly person in a smart home environment," *IEEE Trans. Inf. Technol. Biomed.*, vol. 16, no. 6, pp. 1274–1286, Nov. 2012.

[8] E. Akagündüz, M. Aslan, A. Şengür, H. Wang, and M. C. İnce, "Silhouette orientation volumes for efficient fall detection in depth videos," *IEEE J. Biomed. Health Inform.*, vol. 21, no. 3, pp. 756–763, May 2017.

[9] G. Chen et al., "Neuromorphic vision-based fall localization in event streams with temporal-spatial attention weighted network," *IEEE Trans. Cybern.*, vol. 52, no. 9, pp. 9251–9262, Sep. 2022.

[10] K. Krishnapriya, V. Albiero, K. Vangara, M. C. King, and K. W. Bowyer, "Issues related to face recognition accuracy varying based on race and skin tone," *IEEE Trans. Technol. Soc.*, vol. 1, no. 1, pp. 8–20, Mar. 2020.

[11] S. Moulik and S. Majumdar, "FallSense: An automatic fall detection and alarm generation system in IoT-enabled environment," *IEEE Sensors J.*, vol. 19, no. 19, pp. 8452–8459, Oct. 2019.

[12] H. Wang, D. Zhang, Y. Wang, J. Ma, Y. Wang, and S. Li, "RT-fall: A real-time and contactless fall detection system with commodity wifi devices," *IEEE Trans. Mobile Comput.*, vol. 16, no. 2, pp. 511–526, Feb. 2017.

[13] S. Z. Gurbuz and M G. Amin, "Radar-based human-motion recognition with deep learning: Promising applications for indoor monitoring," *IEEE Signal Process. Mag.*, vol. 36, no. 4, pp. 16–28, Jul. 2019.

[14] M. Forouzanfar, M. Mabrouk, S. Rajan, M. Bolic, H. R. Dajani, and V. Z. Groza, "Event recognition for contactless activity monitoring using phase-modulated continuous wave radar," *IEEE Trans. Biomed. Eng.*, vol. 64, no. 2, pp. 479–491, Feb. 2016.

[15] C. Ding et al., "Non-contact human motion recognition based on UWB radar," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 2, pp. 306–315, Jun. 2018.

[16] B. Jokanović and M. Amin, "Fall detection using deep learning in range-Doppler radars," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 54, no. 1, pp. 180–189, Feb. 2018.

[17] M. S. Seyfioğlu, A. M. Özbayoğlu, and S. Z. Gürbüz, "Deep convolutional autoencoder for radar-based classification of similar aided and unaided human activities," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 54, no. 4, pp. 1709–1723, Aug. 2018.

[18] H. Sadreazami, M. Bolic, and S. Rajan, "Fall detection using standoff radar-based sensing and deep convolutional neural network," IEEE Trans. Circuits Syst. II: Exp. Brief., vol. 67, no. 1, pp. 197–201, Jan. 2020.

[19] H. Li, A. Shrestha, H. Heidari, F. Le Kernec, and F. Fioranelli, "Bi-LSTM network for multimodal continuous human activity recognition and fall detection," *IEEE Sensors J.*, vol. 20, no. 3, pp. 1191–1201, Feb. 2020.

[20] A. Shrestha, H. Li, J. Le Kernec, and F. Fioranelli, "Continuous human activity classification from FMCW radar with Bi-LSTM networks," *IEEE Sensors J.*, vol. 20, no. 22, pp. 13607–13619, Nov. 2020.

[21] B. Erol, "Motion classification using kinematically sifted ACGAN-synthesized radar micro-Doppler signatures," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 4, pp. 3197–3213, Aug. 2020.

[22] H. Sadreazami, M. Bolic, and S. Rajan, "Contactless fall detection using time-frequency analysis and convolutional neural networks," *IEEE Trans. Ind. Inform.*, vol. 17, no. 10, pp. 6842–6851, Oct. 2021.

[23] M. He et al., "Optimum target range bin selection method for time-frequency analysis to detect falls using wideband radar and a lightweight network," *Biomed. Signal Process. Control*, vol. 77, 2022, Art. no. 103741.

[24] Infineon, Neubiberg, Germany, "Position2go radar," Accessed: Jun. 15, 2022. [Online]. Available: https://www.infineon.com/cms/en/product/evaluation-boards/demo-position2go/

[25] M. A. Richards, *Fundamentals of Radar Signal Processing*, 2nd ed. New York, NY, USA: McGraw-Hill, 2014.

[26] E. Tavanti, A. Rizik, A. Fedeli, D. D. Caviglia, and A. Randazzo, "A short-range FMCW radar-based approach for multi-target human-vehicle detection," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 2003816.

[27] C. Shorten et al., "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, 2019, Art. no. 60.

[28] Y. Kim, I. Alnujaim, S. You, and B. J. Jeong, "Human detection based on time-varying signature on range-Doppler diagram using deep neural networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 18, no. 3, pp. 426–430, Mar. 2021.

[29] Google TensorFlow, "Time distributed layer," Accessed: Jun. 15, 2022. [Online]. Available: https://keras.io/api/layers/recurrent_layers/time_distributed/

[30] A. Rizik, E. Tavanti, H. Chible, D. D. Caviglia, and A. Randazzo, "Cost-efficient FMCW radar for multi-target classification in security gate monitoring," *IEEE Sensors J.*, vol. 21, no. 18, pp. 20447–20461, Sep. 2021.

[31] A. Chattopadhay et al., "Grad-CAM : Generalized gradient-based visual explanations for deep convolutional networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2018, pp. 839–847.

[32] M. Sandler et al., "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.

[33] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1251–1258.

[34] K. He et al., "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[35] J. Zhu, X. Lou, and W. Ye, "Lightweight deep learning model in mobile-edge computing for radar-based human activity recognition," *IEEE Internet Things J.*, vol. 8, no. 15, pp. 12350–12359, Aug. 2021.

[36] 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.

[37] G. Lai et al., "Radar-based human activity recognition with 1-D dense attention network," *IEEE Geosci. Remote Sens. Lett.*, vol. 19, pp. 1–5, 2021.

[38] J. Zhu et al., "A hybrid CNN–LSTM network for the classification of human activities based on micro-Doppler radar," *IEEE Access*, vol. 8, pp. 24713–24720, 2020.

[39] Y. Qian, C. Chen, L. Tang, Y. Jia, and G. Cui, "Parallel LSTM-CNN network with radar multi-spectrogram for human activity recognition," *IEEE Sensors J.*, vol. 23, no. 2, pp. 1308–1317, Jan. 2023.

[40] Z. Ou and W. Ye, "Lightweight deep learning model for radar-based fall detection with metric learning," *IEEE Internet Things J.*, vol. 10, no. 9, pp. 8111–8122, May 2023.

[41] C. Ding et al., "Continuous human motion recognition with a dynamic range-Doppler trajectory method based on FMCW radar," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 9, pp. 6821–6831, Sep. 2019.

[42] M. S. Seyfioglu, B. Erol, S. Z. Gurbuz, and M. G. Amin, "DNN transfer learning from diversified micro-Doppler for motion classification," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 55, no. 5, pp. 2164–2180, Oct. 2019.

[43] H. Chen and W. Ye, "Classification of human activity based on radar signal using 1-D convolutional neural network," *IEEE Geosci. Remote Sens. Lett.*, vol. 17, no. 7, pp. 1178–1182, Jul. 2019.

[44] X. Li, Y. He, F. Fioranelli, and X. Jing, "Semisupervised human activity recognition with radar micro-Doppler signatures," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2021, Art. no. 5103112.

[45] Z. Sun, Q. Ke, H. Rahmani, M. Bennamoun, G. Wang, and J. Liu, "Human action recognition from various data modalities: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 3, pp. 3200–3225, Mar. 2023.

[46] Z. Chen, C. Jiang, and L. Xie, "A novel ensemble ELM for human activity recognition using smartphone sensors," *IEEE Trans. Ind. Inform.*, vol. 15, no. 5, pp. 2691–2699, May 2019.

[47] J.-S. Lee and H. H. Tseng, "Development of an enhanced threshold-based fall detection system using smartphones with built-in accelerometers," *IEEE Sensors J.*, vol. 19, no. 18, pp. 8293–8302, Sep. 2019.

[48] F. Zhao, Z. Cao, Y. Xiao, J. Mao, and J. Yuan, "Real-time detection of fall from bed using a single depth camera," *IEEE Trans. Automat. Sci. Eng.*, vol. 16, no. 3, pp. 1018–1032, Jul. 2019.

[49] A. Kamel, B. Sheng, P. Yang, P. Li, R. Shen, and D. D. Feng, "Deep convolutional neural networks for human action recognition using depth maps and postures," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 49, no. 9, pp. 1806–1819, Sep. 2019.

[50] H. Ramirez et al., "Fall detection and activity recognition using human skeleton features," *IEEE Access*, vol. 9, pp. 33532–33542, 2021.

Christian Gianoglio (Member, IEEE) received the master's degree in electronic engineering and the Ph.D. degree in electrical engineering from the University of Genoa, Genoa, Italy, in 2015 and 2018, respectively.

He is currently a Research Fellow with DITEN, University of Genoa. His research interests include machine learning for resource-constrained devices and real-time applications, signal processing and machine learning for tactile sensing systems, and pattern recognition for quality assessment of insulation systems in electrical apparatuses.

Ammar Mohanna received the master's degree in software engineering from the Saint Joseph University of Beirut, Beirut, Lebanon, in 2019, and the Ph.D. degree in science and technology for electronic and telecommunication engineering from tthe University of Genoa, Genoa, Italy, in 2023.

He currently serves as the AI Lead with Assentify, Beirut, Lebanon, where his focus lies in implementing AI-related microservices within the domains of Digitalization and Insurance. He is also a university Lecturer, specializing in courses pertaining to AI and MLOps.

Ali Rizik received the master's degree in signal, telecom, image, and speech processing from the Faculty of Science, Lebanese University of Beirut, Beirut, Lebanon, in 2016, and the Ph.D. degree in science and technology for electronic and telecommunication engineering from University of Genoa, Genoa, Italy, in 2021.

He currently occupies a Postdoctoral position with DIATI, Politecnico di Torino, Turin, Italy, contributing to the WIVERN spaceborne Doppler radar project. His research interests include a broad spectrum, with a primary focus on machine learning applications, radar signal processing, radar target identification for security applications, and the application of radars in meteorology.

Laurence Moroney received the B.Sc. degree in physics and computer science from Cardiff University, Cardiff, U.K., in 1991, the Ph.D. degree in microelectronics system design with Birmingham City University, Birmingham, U.K., in 1993, and a graduate certificate in AI from Standford University, Stanford, CA, USA, in 2018.

He is currently the AI Advocacy lead with Google, Mountain View, CA, USA, where he likes to inform and inspire the world at scale about the possibilities of AI. He is an award-winning Researcher, teacher of millions of students in online classes, creator of popular datasets, and massively enthusiastic about what we can do with AI and ML when approaching it in an educated way. He has authored or coauthored works which include many best-selling books.

Maurizio Valle (Senior Member, IEEE) received the M.S. degree in electronic engineering, and the Ph.D. degree in electronics and computer science from the University of Genoa, in 1985, and 1990, respectively.

Since December 2019, he has been a Full Professor of Electronics with DITEN, University of Genoa, where he leads the Connected Objects, Smart Materials, Integrated Circuits Laboratory. He has been and is in charge of many research contracts and projects funded at local, national, and European levels and by Italian and foreign companies. His research interests include biomedical circuits and systems, electronic/artificial sensitive skin, tactile sensing systems for prosthetics and robotics, neuromorphic touch sensors, and electronic, and microelectronic systems.