TinyML for UWB-radar based presence detection

Massimo Pavan
Politecnico di Milano
Milan, Italy
massimo.pavan@polimi.it

Armando Caltabiano
Truesense s.r.l.
Milan, Italy
armando.caltabiano@truesense.it

Manuel Roveri
Politecnico di Milano
Milan, Italy
manuel.roveri@polimi.it

***Abstract*—Tiny Machine Learning (TinyML) is a novel research area aiming at designing machine and deep learning models and algorithms able to be executed on tiny devices such as Internet-of-Things units, edge devices or embedded systems. In this paper we introduce, for the first time in the literature, a TinyML solution for presence-detection based on UltrawideBand (UWB) radar, which is a particularly promising radar technology for pervasive systems. To achieve this goal we introduce a novel family of tiny convolutional neural networks for the processing of UWB-radar data characterized by a reduced memory footprint and computational demand so as to satisfy the severe technological constraints of tiny devices. From this technological perspective, UWB-radars are particularly relevant in the presence-detection scenario since they do not acquire sensitive information of users (e.g., images, videos or audio), hence preserving their privacy.**

The proposed solution has been successfully tested on a public-available benchmark for the indoor presence detection and on a real-world application of in-car presence detection.

***Index Terms*—Tiny Machine Learning, UltraWideBand (UWB) radar, Presence detection, Privacy-preserving computation.**

I. Introduction

In recent years the technological evolution is paving the way for a pervasive diffusion of tiny devices, such as Internet-of-Things (IoT) units, edge devices and embedded systems, representing the technological asset of the "computing everywhere" paradigm [1][2]. From this technological perspective, the scientific trend is to move the processing (and in particular the intelligent processing) as close as possible to where data are generated to increase the autonomy of tiny devices, reduce the latency with which a decision is made, reduce the required transmission bandwidth and increase the energy efficiency [3][4]. Designing Machine and Deep Learning solutions (MDL) able to be executed on these tiny devices requires to completely re-think and re-design MDL models and algorithms so as to take into account the severe constraints on memory (the available RAM is in the order of the MB), computation (the MCU frequency is in the order of the MHz), and power consumption (typically < 0.1 W) of these devices.

This is exactly where Tiny Machine Learning (TinyML) comes into play by designing, developing and deploying MDL models and algorithms for tiny devices. TinyML solutions present in the literature typically introduce tiny MDL architectures (characterized by reduced memory and computational demands of the processing layers and weights) and approximate-computing solutions (such as quantization [5], pruning[6], and early-exit mechanisms[7][8]) to fit the severe technical constraints characterizing these tiny devices.

Interestingly, one of the most promising application scenarios of such intelligent tiny devices is "presence detection", i.e., detecting the presence of a person in a given environment. Currently the research in the field of TinyML for presence detection focused on the analysis of data coming from cameras (formalized as a "visual wake-word detection" problem) or microphones (formalized as a "keyword/sound spotting" problem) [9][10]. Unfortunately, acquiring and processing these types of data pose serious and legitimate concerns in terms of user privacy since images, videos or audio of people can be considered sensitive information [11].

To address this challenging and relevant issue we introduce here, for the first time in the literature, the use of radar sensors in TinyML for presence detection. In more detail, we focus on UltrawideBand (UWB) radar, which is a particularly promising radar technology for tiny devices. Indeed uwb-radars are characterized by high precise recordings (they can detect changes in the environment in the order of the mm), low energy consumption (typically < 0.1 W) and fast acquisition of data (each scan requires only some fraction of seconds to be collected) [12]. We emphasize that enabling the use of TinyML solutions for UWB-radar will pave the way for intelligent pervasive applications based on tiny devices enforcing the privacy of the users.

The aim of this paper is to introduce a new family of Tiny Convolutional Neural Networks (TyCNNs) for presence-detection through UWB-radar data on tiny devices. This family of CNNs relies on an integrated ad-hoc design of tiny dilated convolutional blocks and quantization of the CNN architecture to reduce the computational and memory demands (of both weights and activations). The proposed family of CNNs is complemented with a suitably-defined pre-processing phase to further reduce the memory and computational demand, while maintaining the presence-detection accuracy.

The proposed solution has been successfully tested on both a public-available benchmark for UWB-radar-based presence detection and a real-world in-car presence detection application. In particular, the proposed UWB-based TinyML solution for the in-car presence-detection has been successfully deployed and tested in real-world conditions on an ESP32 microcontroller unit (4 MB of flash memory, 512 KB of S-RAM memory), equipped with a uwb-radar module comprising only one pair of antennas.

The paper is organized as follows. Section II describes the related literature, while Section III introduces the proposed TinyML solution for UWB-based presence-detection. Section IV details the problem definitions and the experimental results for the indoor and in-car presence detection scenarios. Finally, Section V draws the conclusion and describes the future research directions in this field.

II. Related Literature

This section describes the related literature in the field of TinyML (Section II-A) and the available UWB-radar solutions for presence detection and activity recognition (Section II-B). We emphasize that, currently, no TinyML solution able to process UWB-radar data is available in the literature.

A. TinyML

The research in the field of MDL for embedded systems and IoT units is mainly addressed from two different perspectives: the development of custom hardware and the design of approximated MDL solutions.

In the last few years, the development of *custom hardware* specifically designed to run MDL solutions has radically improved the performance and reduced power consumption. Unfortunately, these advantages come at the expense of an increased complexity of the design phase along with a lower flexibility with respect to general-purpose hardware [13]. In such a technological scenario [14], GPUs , TPUs, or neural hardware would allow to significantly improve the overall performances in terms of training and inference time, but their power consumption makes them not a viable solution for embedded systems and IoT units.

The design of *approximated machine/deep learning solutions* capable of addressing the strict technological constraints of embedded and IoT units (in terms of memory, computation and energy) is a relevant and continuously-growing research field. Despite being a very fragmented area, the techniques introduced in this area can generally fall within the field of TinyML [10][15]. Most of the related literature focuses on the approximation of Convolutional Neural Network algorithms. For example, [16] introduced a methodology to explore sparse CNN architectures that could be executed on Microcontroller units (MCUs), whereas [17] proposed Bonsai, a decision tree-based technique to perform CNN-inference efficiently on Arduino boards. In addition, pruning of channels and layers of CNNs has proven to be a successful [18][19] in reducing the memory and computational demand.

A different approach to approximate CNNs aims at reducing the memory required to store the CNN weights through quantization by using limited-precision data types [20][21] or binary weights [22][23]. In such a direction, [24] combined both task dropping and precision scaling techniques to design approximated CNNs able to be executed in IoT units.

Other solutions focus on reducing the mean inference time of deep neural networks. In this direction, Adaptive Early Exit [8] and Gate-Classification CNNs [7] can provide the final classification output even at intermediate layers according to the input content, thus not always requiring the execution of the whole CNN pipeline. However, in these architectures the inference time is typically reduced at the expense of an increase in the memory footprint.

In the field of TinyML for presence detection, [25] provided a dataset for visual wake-word tasks (here presence-detection is considered one of the most interesting visual wake-work tasks) and a presence-detection solution based on the analysis of images is also suggested. Similarly, [26] provided a dataset for “keyword detection”, which is one of the most successful tasks for TinyML. The analysis of audio data can be in principle used to detect the presence of people talking in an environment (e.g., by speaking to a vocal assistant or to a smart switch in a smart city).

B. UWB-radar for presence detection and activity recognition

The literature about UWB-radar solutions for human detection is quite wide and comprises person detection and human activity recognition (HAR). In this related literature, to ease the comparison, we focus our attention on UWB-radar solutions that rely on a single receiving antenna[1].

Presence detection: Most of the solutions for presence detection based on uwb-radar relies on thresholds or statistical approaches to distinguish between empty records and records where a human is present [27][28]. The main drawback of these solutions is that the detection ability is strictly related to the noise of the sensor and to the environment in which the sensor is deployed (hence limiting the generalization ability of the solution in different environments or with different types of radar). Few UWB-based solutions implementing MDL algorithms are available in the literature for this task. In [29] for example, deep convolutional neural networks have been successfully used by exploiting the micro-doppler representation of the radar data. Finally, anti-abandon systems for cars based on radar can be also found in the literature [30][31]. Unfortunately, these solutions do not rely on uwb-radar and they are only meant to detect the presence of a child in a car (they cannot generalize to young or adult presence detection).

Human Activity recognition: Differently from UWB-based person detection solutions, most of the HAR solutions based on UWB present in the literature focuses on machine learning solutions and, in particular, on CNNs architectures [32]. Such solutions can be grouped into two main families depending on how the training is carried out. The first family of solutions [33][34] relies on transfer-learning mechanisms to classify the spectrograms of the radar data, while the second one [35] aims at designing CNNs from scratch. In the aforementioned solutions, time-doppler maps [36] are typically used as inputs to the networks, since they proved to be very effective to preprocess radar data both with and without ML solutions [37].

[1]Each solution is characterized by its own preprocessing phase: no transformations, transformation in range-doppler or in time-doppler maps.

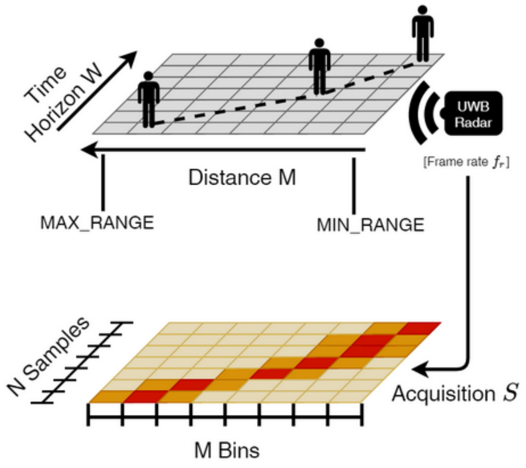

Fig. 1. The acquisition of matrix S by the UWB-radar antenna.

Unfortunately, the UWB-based solutions for HAR present in the literature are not a viable technological solution on tiny devices due to the high memory and computational demands.

Finally, we acknowledge the presence in the literature of an edge solution [38] that makes use of UWB-radar data to recognize hand gestures through the implementation of a deep learning algorithm. Nevertheless, specialized hardware (Intel Neural Computer Stick - NCS2) was used for the deep learning part of this research, and thus it cannot be considered a TinyML solution.

III. The proposed TinyML solution for UWB-radar based presence detection

The proposed TinyML solution for presence detection based on UWB-radar comprises two main modules: pre-processing and tiny deep convolutional neural network. These two modules, which are detailed in the sequel, have been jointly designed and developed to maximize the presence-detection accuracy, while satisfying the strict technological constraints of tiny devices.

A. The pre-processing module

The aim of the pre-processing module is to highlight the relevant information present in the acquired UWB-radar data and remove the noise. The proposed pre-processing module for the tiny UWB-radar based presence detection has been carefully designed to minimize the computational and memory demands, hence minimizing its overhead in the target tiny device. Details about memory and computational demands of the preprocessing module are reported in Section IV.

Let $S \in \mathbb{R}^{N \times M}$, with $M, N \in \mathbb{N}$, be the output of the UWB-radar receiving-antenna installed on the device, being N the collected number of radar scans and M the number of "bins" characterizing the acquisitions of the antenna. In more detail, the value $S[i, j]$ with $i = \{1, \ldots, N\}$ and $j = \{1, \ldots, M\}$ represents the energy acquired by the i-th scan at the j-th bin. We emphasize that $N = W \cdot f_r$, being f_r the UWB-radar frame rate (i.e., the number of acquisitions per seconds) and W the acquisition time horizon (in seconds), while M represents the number of "quantized" distances in the acquisition range, i.e., from MIN_RANGE to MAX_RANGE, of the UWB-radar antenna[2]. An example of the acquisition of S is shown in Figure 1.

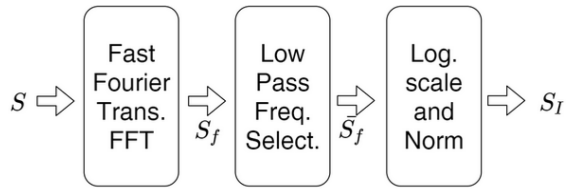

Fig. 2. The preprocessing steps.

As shown in Figure 2, the pre-processing module comprises the following three steps: Fast-Fourier Transform, low-pass frequency selection and data normalization.

1) Fast Fourier Transform: The first step of pre-processing aims at computing the Fast Fourier Transform (FFT) S_f of S. The considered FFT algorithm is the Cooley-Tukey algorithm [39]. The FFT is computed on all the M rows of S. For this purpose, without any loss of generality[3], we assume N being a power 2 value.

The considered FFT computes both positive and negative frequencies. We focus only on the positive frequencies since negative frequencies contain the conjugate of the positives, being S a matrix of *real* numbers. For this reason the output of this step is $S_f \in \mathbb{R}^{\frac{N}{2} \times M}$ and the $\frac{N}{2}$-th column represents the Nyquist frequency $f_r/2$.

2) Low-pass frequency selection: The aim of this step is to select only a sub-range of the frequencies in S_f to reduce the memory and computational demands of the next tiny convolutional neural network (see Section III-B). In more detail, let $f_l < f_r/2$ be the selected cut-off frequency, the goal of this module is to remove from S_f the rows corresponding to the frequencies larger than f_l. Defining f_l is typically application-specific. In the domain of person detection, the relevant information lies usually in the range $0 - 25$Hz but f_l can be scaled down (with a negligible loss of accuracy) to 2Hz (or less) to further reduce the memory and computational demands of the tiny convolutional neural network.

Let L be the index of the row in S_f corresponding to f_l, the output of the low-pass frequency selection module is $\bar{S}_f \in \mathbb{R}^{L \times M}$. In Section IV we introduce two real-world presence detection scenarios (i.e., indoor and in-car) where f_s has been set to 25Hz and 1.66Hz, respectively.

3) Data normalization: Once the FFT of the acquired UWB-data has been computed and the relevant frequencies have been selected, data in $\bar{S}_f$ are finally processed by means of a log-scale transformation and a Z-score normalization

[2]Each value represents the amount of energy of the reflected radar wave. MIN_RANGE, MAX_RANGE and M are parameters depending on the specific radar device used and on its configuration.

[3]When N is not power-of-two we can consider padding mechanisms and the use of the Hann window [40].

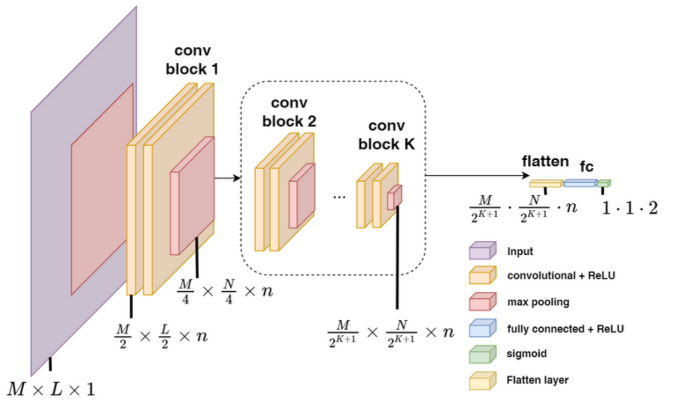
conv block 1
conv block 2
conv block K
flatten
fc
$\frac{M}{2^{K+1}} \cdot \frac{N}{2^{K+1}} \cdot n$
$1 \cdot 1 \cdot 2$
Input
convolutional + ReLU
max pooling
fully connected + ReLU
sigmoid
Flatten layer
$\frac{M}{4} \times \frac{N}{4} \times n$
$\frac{M}{2} \times \frac{L}{2} \times n$
$\frac{M}{2^{K+1}} \times \frac{N}{2^{K+1}} \times n$
$M \times L \times 1$

Fig. 3. Representation of the sizes of the activations of the tiny convolutional neural network TyCNNs.

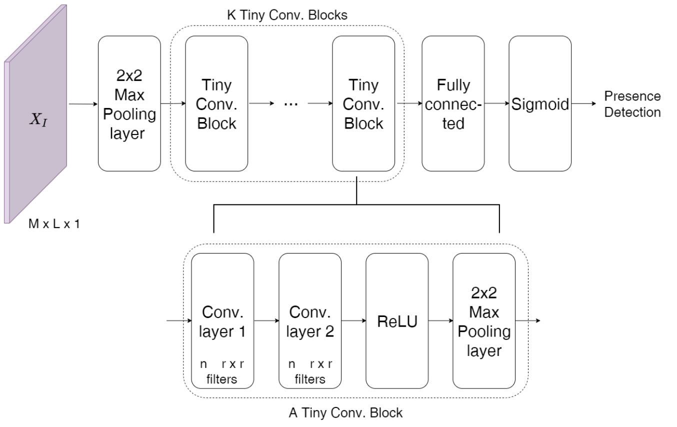

Fig. 4. The general architecture of the tiny convolutional neural networks TyCNNs.

(where mean and standard deviation are computed on the training set of the tiny convolutional neural networks).

The output of this module is $S_I \in \mathbb{R}^{L\times M}$ and represents the input of the next tiny convolutional neural network.

B. Tiny convolutional neural networks for UWB-radar

We here introduce a family of tiny convolutional neural networks (TyCNNs) for presence detection with UWB-radar data. These TyCNNs have been carefully designed to satisfy the technological constraints on the memory footprint $\bar{m}$ (in KB) and computation $\bar{c}$ (number of operations) of the target tiny devices. To achieve this goal, TyCNNs rely on:

- a suitably-defined CNN architecture characterized by a tunable number of ad-hoc dilated convolutional blocks and by a final fully connected layer (integrating a sigmoid activation function);
- a post-training quantization mechanism to transform the TyCNN weights and activations from 32-bit floating point to 8-bit integer. The quantization is also applied to the input S_I leading to $X_I \in \mathbb{Z}^{L\times M}$.

An overview of the proposed TyCNNs is shown in Figures 3 and 4, while the detailed TyCNN architecture and the quantization mechanism are described in what follows.

Being X_I the input of the TyCNN, the processing layers can be summarized as follows:

1) A 2×2 Max Pooling layer: this layer aims at reducing the size of the input X_I. In more detail, the goal of this layer is to reduce the memory demand of intermediate activations as well as the number of operations required by the TyCNNs to compute the inference. We emphasize that scaling down X_I allows also to reduce the memory demand of the final fully connected layer, while it does not have any impact on the memory demand of the weights of the convolutional layers.

2) A sequence of K Tiny Convolutional Blocks: the Tiny Convolutional Blocks (TCBs) represent the core of the TyCNNs architecture. Each block comprises the four following steps:

- a first convolutional layer comprising n square $r\times r$ dilated filters with dilation rate equal to 2;
- a second convolutional layer comprising n square $r\times r$ dilated filters with dilation rate equal to 2;
- the ReLu activation function;
- a 2×2 Max Pooling layer.

It is worth noting that the first two steps of the TCBs share the same configuration of the filters. We emphasize that, for these two steps, we considered dilated filters [41] since they have proven to be effective in enlarging the field-of-reach of the filter, while not increasing the number of weights (hence not increasing the memory demand associated with the TyCNN weights). In the considered presence-detection applications described in Section IV, the following configuration of the TBCs have been considered: $n = 14$ and $r = 5$.

3) A fully-connected layer: The aim of this last layer is to provide the final classification of the TyCNN. In more detail, this layer comprises a flattening layer, a dropout layer (with dropout rate equals to 0.3), and a single dense layer with sigmoid activation characterized by two outputs (since we modelled the presence-detection as a binary classification problem, i.e., present vs. not present).

It is worth noting that training and quantization are strictly related in TyCNNs. For the training we considered the Binary Crossentropy as loss function, while we used Adam as optimizer. The learning rate was set to 0.3e-4, while the number of training epochs was set to 15 and 400 for indoor and in-car presence detection, respectively. Once the TyCNN has been trained, the full-integer post-training weight quantization algorithm introduced in [42] has been used to transform the 32-bit floating-point weights into 8-bit integers. The considered post-training quantization algorithm has been also applied to inputs and activations to guarantee that the values of both network weights and activations are stored as 8-bit integers. Besides reducing the memory, the quantization allows also the use of integer operations on the target tiny device, which are faster and less energy-hungry than the corresponding 32-bit floating-point operations.

C. Evaluating the memory footprint and computational load of TyCNNs

Evaluating the memory footprint m and the computational load c of TyCNNs at design-time is crucial to support their porting on tiny devices characterized by constraints on memory $\overline{m}$ and computation $\overline{c}$. For this purpose we extend here the formalization introduced in [24], by defining m and c as follows:

$$\begin{aligned} m &= (\hat{m_w} + \hat{m_a}) \cdot m_p, \\ c &= n_{ops} \end{aligned}$$

being $\hat{m_w}$ the cardinality of the parameters of the TyCNN, n_{ops} the number of multiplications required to compute the inference, $\hat{m_a}$ the memory needed to store the maximum sum of two consecutive activations, and m_p the memory required to store a value of a parameter or an activation. In our case m_p is equal to 1 Byte (8 bit) since both the weights of the network and the input data are quantized.

We emphasize that, since arrays used to store intermediate activations are shared among the processing layers of the network and activations' size can be calculated in advance for every layer, $\hat{m_a}$ can be easily evaluated at design time. In more detail, being L, M, n, K the parameters described in Section III, it is easy to demonstrate that:

$$\hat{m_a} = \begin{cases} (\frac{L}{2} \cdot \frac{M}{2}) \cdot n \cdot 2 & K \geq 3 \\ (\frac{L}{2} \cdot \frac{M}{2}) + M \cdot L & K < 3 \end{cases}.$$

Differently $\hat{m_w}$ and n_{ops} can be computed as:

$$\begin{aligned} \hat{m_w} &= \sum_{X=0}^{K} m_w^X + \sum_{C=1}^{2K} m_w^C + m_w^{FC} \\ n_{ops} &= \sum_{X=0}^{K} n_{ops}^X + \sum_{C=1}^{2K} n_{ops}^C + n_{ops}^{FC} \end{aligned}$$

where m_w^C, m_a^C, n_{ops}^C represent the number of Bytes needed to store the weights, the number of Bytes needed to store the activations and the total number of operations for the convolutional layers, respectively, m_w^X, m_a^X, n_{ops}^X represent the number of Bytes needed to store the weights, the number of Bytes needed to store the activations and the total number of operations for the Max Pooling layers, respectively, and m_w^{FC}, m_a^{FC}, n_{ops}^{FC} represent the number of Bytes needed to store the weights, the number of Bytes needed to store the activations and the total number of operations for the fully connected layers, respectively.

In more detail, for each convolutional layer of the network, we can compute m_w^C, m_a^C and n_{ops}^C as follows:

$$\begin{aligned} m_w^C &= (r^2 \cdot n_{in} \cdot n + n) \cdot m_p \\ m_a^C &= (L_{in} \cdot M_{in} \cdot n) \cdot m_p \\ n_{ops}^C &= r^2 \cdot n_{in} \cdot n \cdot L_{in} \cdot M_{in} \end{aligned}$$

while L_{in}, M_{in} and n_{in} are the output dimensions of the layer preceding the convolutional one.

TABLE I
THE DETAILED MEMORY FOOTPRINT AND THE COMPUTATIONAL DEMAND OF THE TYCNN.

	Memory Footprint (B)	N. operations
S_I	$M \cdot L \cdot 1$	-
Pool0 (Weights)	-	-
S_I–Pool0 (Activations)	$\frac{L}{2} \cdot \frac{M}{2} \cdot 1$	$2 \cdot 2 \cdot L \cdot M$
Conv1_00 (Weights)	$r^2 \cdot n + n$	-
Conv1_00 (Activations)	$\frac{L}{2} \cdot \frac{M}{2} \cdot n$	$r^2 \cdot n \cdot \frac{L}{2} \cdot \frac{M}{2}$
Conv1_01 (Weights)	$r^2 \cdot n^2 + n$	-
Conv1_01 (Activations)	$\frac{L}{2} \cdot \frac{M}{2} \cdot n$	$r^2 \cdot n^2 \cdot \frac{L}{2} \cdot \frac{M}{2}$
Pool1 (Weights)	-	-
Pool1 (activations)	$\frac{L}{4} \cdot \frac{M}{4} \cdot n$	$2 \cdot 2 \cdot \frac{L}{2} \cdot \frac{M}{2}$
ConvK_00 (Weights)	$r^2 \cdot n^2 + n$	-
ConvK_00 (Activations)	$\frac{L}{2K} \cdot \frac{M}{2K} \cdot n$	$r^2 \cdot n^2 \cdot \frac{L}{2K} \cdot \frac{M}{2K}$
ConvK_01 (Weights)	$r^2 \cdot n^2 + n$	-
ConvK_01 (Activations)	$\frac{L}{2K} \cdot \frac{M}{2K} \cdot n$	$r^2 \cdot n^2 \cdot \frac{L}{2K} \cdot \frac{M}{2K}$
PoolK (Weights)	-	-
PoolK (activations)	$\frac{L}{2K+1} \cdot \frac{M}{2K+1} \cdot n$	$2 \cdot 2 \cdot \frac{L}{2K+1} \cdot \frac{M}{2K+1}$
FC Classifier (Weights)	$\frac{L}{2K+1} \cdot \frac{M}{2K+1} \cdot n \cdot 2 + 2$	-
FC Classifier (activations)	2	$\frac{L}{2K+1} \cdot \frac{M}{2K+1} \cdot n \cdot 2$

Differently, for each Max Pooling layer of the network, we can compute m_w^X, m_a^X and n_{ops}^X as follows:

$$\begin{aligned} m_w^X &= 0 \\ m_a^X &= (\frac{L_{in}}{2} \cdot \frac{M_{in}}{2} \cdot n_{in}) \cdot m_p \\ n_{ops}^X &= 2 \cdot 2 \cdot L_{in} \cdot M_{in} \end{aligned}$$

Finally, for the Fully Connected layer, m_w^X, m_a^X and n_{ops}^X can be computed as follows:

$$\begin{aligned} m_w^{FC} &= (L_{in} \cdot M_{in} \cdot n \cdot 2 + 2) \cdot m_p \\ m_a^{FC} &= 2 \cdot m_p \\ n_{ops}^{FC} &= 2 \cdot n \cdot L_{in} \cdot M_{in} \end{aligned}$$

Table I details the required memory in terms of weights and activations, along with the total number of operations for all the layers of TyCNN, computed with the formulas described above. Tables III and IV will detail the specific values of memory footprint and number of operations for the two application scenarios presented in this paper.

IV. EXPERIMENTAL RESULTS: INDOOR AND IN-CAR PRESENCE DETECTION

The proposed TinyML UWB-based solution has been tested in two relevant and challenging presence-detection scenarios: indoor and in-car. In more detail, Section IV-A describes the application of the proposed solution in an indoor presence detection scenario. For this purpose we considered a public-available dataset present in the literature. Differently, Section IV-B introduces the in-car presence detection scenario. This is a particularly relevant scenario since the possibility to use a tiny device equipped with an UWB-radar to detect the presence of sensible targets in the rear seats of a car could lead to the development of privacy-preserving non-invasive safety devices.

For the in-car presence detection scenario, we considered a target tiny device based on the ESP32 microcontroller unit, we designed and carried out an experimental data-acquisition campaign on real cars and real people and we finally ported the proposed solution on the target tiny device.

A. Indoor presence detection

1) Problem definition: This problem can be formalized as a binary classification task aiming at discriminating whether a person is present or not in the recording. The person can be moving or standing in front of the radar device.

2) Data description and preprocessing: For this experimental campaign we considered data from the IR-UWB-Through-wall-Radar-Human-Motion-Status-Dataset [43]. In more detail, data belonging to Class 001 (human walking) and 010 (standing still) of the dataset have been assigned to the positive Class 1 (presence), while the data belonging to Class 100 (empty) to the negative Class 0. Both "through-wall" and "through-air" data of the dataset have been used. The dataset provides data already partitioned into training and test set. We kept this organization in order to be able to compare our solution with the one proposed in [43].

In this scenario S is characterized by M = 32 and N = 768, and it represents radar scans of 4 seconds acquired with a frame rate f_r = 192 Hz. The data have been preprocessed following the steps described in Section III-A. f_l has been set to 25 Hz, such that S_I, used as input for the network, is characterized by M = 32 and L = 100. The TyCNN used in this scenario relies only on one TCB, i.e. K = 1.

3) Experimental results: In this section we report the results provided by our TyCNN on the test set and compare them to the results obtained by the classification algorithm proposed in [43]. Experimental results are given in Table II in terms of Accuracy, Precision, Recall and F2-score, along with the memory footprint m and the total number of operations c.

Interestingly, the proposed solution provides classification abilities that are in line with those of [43] (i.e., the reduction in the classification accuracy is about 0.5%). Conversely, the proposed solution guarantees a 99.93% reduction of the memory footprint m, and a 99.94% reduction of the total number of operations c with respect to the algorithm proposed in [43]. We emphasize that the solution in [43] cannot be considered a viable algorithmic solution for a tiny device due to the high memory and computational demands. The list of the memory footprint and the number of operations divided per layer for the proposed TyCNN solution are detailed in Table III, while its confusion matrix for the presence-detection problem is here reported:

True \Pred	not present (0)	present (1)
not present (0)	273 (98.9 %)	3 (1.1 %)
present (1)	1 (0.2 %)	551 (99.8 %)

B. In-car presence detection

1) Problem definition: This problem can be formalized as a binary classification task aiming at discriminating whether a subject (adult or child) is present in any of the rear seats of the car.

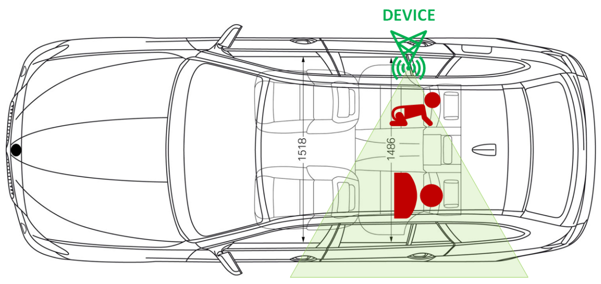

Fig. 5. The acquisition campaign for this experimental analysis

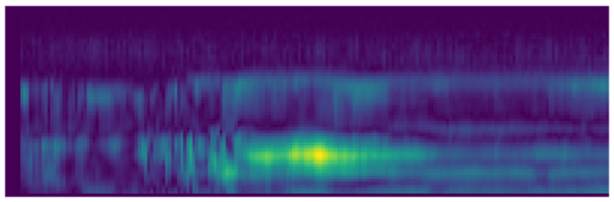

Fig. 6. An example of acquisition for the in-car scenario with M=53 and N=200.

2) The target device: The considered tiny device is based on an ESP32 Microcontroller unit (MCU). This MCU is equipped with both a Wi-Fi (used mainly in the development and testing phase) and a Bluetooth module. Furthermore, the device can rely on a 4MB Flash memory, and two RAM memories (520KB SRAM, 16KB SRAM in RTC). Since part of the memory needs to be allocated for the firmware of the device, we considered $\bar{m}$ = 100 KB, and set a limit on the execution time of the algorithm of 1 s. The device also provides an uwb-radar module, equipped with an ultra-high precision radar sensor developed by ARIA sensing (LT103OEM UWB) that relies only on one pair of antennas (one transmitting TX and one receiving RX). The device was used for both collecting the data and deploying the proposed solution for the in-car presence-detection.

3) Data collection: In all the recordings the device was deployed above one of the back lateral windows of the car (the radar can detect subjects in a $\pm 60°$ cone from where it's directed). The total amount of acquired samples is 208, divided into 84 records with a target present and 124 empty records. Figure 5 describes the positioning and the cone of view of the device during the data-collection phase.

In this scenario S is characterized by M = 53, N = 200, and each acquisition lasts W = 20 s, with a frame rate f_r = 10 Hz. The data have been preprocessed following the steps described in section III-A. f_l has been set to 1.66 Hz, such that the dimensions of S_I are M = 53 and L = 86. For this scenario, the number of TCB is set to 3. An example of a recording is reported in Figure 6.

4) Experimental results: For the experimental results the dataset has been randomly split into 85% for the training and 15% for the testing, five runs have been considered and the results averaged (the standard deviation has been also

TABLE II
COMPARISON OF THE RESULTS OF THE TYCNN AND THE RESPECTIVE BASELINE ALGORITHMS. N.V. : NOT VIABLE FOR TINY DEVICES.

Network	Accuracy	Precision	Recall	F1	m (kB)	c (10^6)
baseline DL (indoor)	1	1	1	1	$\approx$ 52550.0 (N.V.)	$\approx$ 1425.0 (N.V.)
TyCNN (indoor)	0.995	0.995	0.998	0.996	32.5	4.2
baseline TL (in-car)	0.925$\pm$0.022	0.935$\pm$0.029	0.876$\pm$0.033	0.905$\pm$0.028	$\approx$ 18300.0 (N.V.)	$\approx$ 1150.0 (N.V.)
TyCNN (in-car)	0.906$\pm$0.055	0.906$\pm$0.056	0.861$\pm$0.116	0.877$\pm$0.078	55.3	9.2

TABLE III
THE DETAILED MEMORY FOOTPRINT (WITH AN 8-BIT DATA TYPE) AND NUMBER OF OPERATIONS OF THE TYCNNS FOR THE INDOOR SCENARIO. TO OPTIMIZE THE MEMORY, TWO ARRAYS ONLY ARE USED TO STORE THE ACTIVATIONS (AN ASTERISK MARKS THE ACTIVATIONS RE-USING SUCH ARRAYS).

	Memory Footprint (B)	N. operations
S_I	$*32 \cdot 100 \cdot 1 = 3200$	-
Pool0 (Weights)	-	-
S_I–Pool0 (Activations)	$*16 \cdot 50 \cdot 1 = 800$	$2 \cdot 2 \cdot 32 \cdot 100 = 12800$
Conv1_00 (Weights)	$25 \cdot 14 + 14 = 364$	-
Conv1_00 (Activations)	$16 \cdot 50 \cdot 14 = 11200$	$25 \cdot 14 \cdot 16 \cdot 50 = 280000$
Conv1_01 (Weights)	4914	-
Conv1_01 (Activations)	11200	3.920.000
Pool1 (Weights)	-	-
Pool1 (activations)	$*2800$	3200
FC Classifier (Weights)	5602	-
FC Classifier (activations)	*2	5600
Total	33280	4221600

TABLE IV
THE DETAILED MEMORY FOOTPRINT (WITH AN 8-BIT DATA TYPE) AND THE NUMBER OF OPERATIONS OF THE TYCNNS FOR THE IN-CAR SCENARIO. TO OPTIMIZE THE MEMORY, TWO ARRAYS ONLY ARE USED TO STORE THE ACTIVATIONS (AN ASTERISK MARKS THE ACTIVATIONS RE-USING SUCH ARRAYS).

	Memory Footprint	10^6 operations
S_I	$*53 \cdot 86 \cdot 1 = 4558$	-
Pool0 (Weights)	-	-
S_I–Pool0 (Activations)	$*26 \cdot 43 \cdot 1 = 1118$	$2 \cdot 2 \cdot 53 \cdot 86 = 18232$
Conv1_00 (Weights)	$25 \cdot 14 + 14 = 364$	-
Conv1_00 (Activations)	$26 \cdot 43 \cdot 14 = 15652$	$25 \cdot 14 \cdot 26 \cdot 43 = 391300$
Conv1_01 (Weights)	4914	-
Conv1_01 (Activations)	15652	5.478.200
Pool1 (Weights)	-	-
Pool1 (activations)	$*3822$	4472
Conv2_00 (Weights)	4914	-
Conv2_00 (Activations)	$*3822$	1337700
Conv2_01 (Weights)	4914	-
Conv2_01 (Activations)	$*3822$	1337700
Pool2 (Weights)	-	-
Pool2 (activations)	$*840$	1092
Conv3_00 (Weights)	4914	-
Conv3_00 (Activations)	$*840$	294000
Conv3_01 (Weights)	4914	-
Conv3_01 (Activations)	$*840$	294000
Pool3 (Weights)	-	-
Pool3 (activations)	$*210$	240
FC Classifier (Weights)	422	-
FC Classifier (activations)	*2	420
Total	56660	9157355

computed).

Table II describes the classification abilities of the proposed solution together with the memory footprint m and the computational load c for the in-car presence detection scenario.

As a comparison we considered a CNN inspired by the MobileNet [44], which is a popular CNN for mobile devices and edge computing systems. In more detail, following a transfer learning approach (due to the limited number of samples we had in our training set), we considered the convolutional part of MobileNet (followed by a global average pooling) as a feature extractor. Then, extracted features are processed by a single Dense layer that performs the classification (this layer is trained on the training set). A dropout layer (dropout rate = 0.2) is also considered to reduce the overfitting. Since radar data comprises only one channel, during the preprocessing the channel was repeated 3 times to match the required input dimension of the network (i.e. the networks requires a 3 channel input). The optimizer for the training was Adam (lr = 10^{-5}), the selected loss Binary Crossentropy and the model was trained for 200 epochs.

It is worth mentioning that, despite the use of the MobileNet, the CNN used for the comparison does not match the constraints on memory and computation (e.g., even just the memory to store the weight of the MobileNet requires up to 17.9 MBs, or 4.2 MBs using quantization). Differently the proposed solution completely matches these technological constrains with $m = 56.6$ and $c = 9.16e6$. We measured experimentally the execution time of the solution on the ESP32 board. The total execution time is 980 ms, divided in 230 ms for preprocessing data and 750 ms to perform the inference with the TyCNN. Preprocessing required 27136 B to be executed in memory, and thus can be executed in the same memory space of dimension $\hat{m_a} = 31304$ B where the activations of the networks will be stored, hence not influencing the memory footprint. The loss in accuracy induced by the proposed solution is however limited (less than 2%) showing not only the efficiency but also the effectiveness of what proposed.

The detailed list of the memory footprints and the number of operations divided by layers for the proposed solution are detailed in Table IV, while the confusion matrix is here reported:

True \Pred	not present (0)	present (1)
not present (0)	89 (93.7%)	6 (6.3%)
present (1)	9 (13.8%)	56 (86.2%)

V. CONCLUSIONS

The aim of this paper was to introduce, for the first time in the literature, a TinyML solution for presence-detection based on UWB-radar. To achieve this goal we introduced a family of tiny convolutional neural networks based on dilated convolutional blocks able to guarantee high detection ability and reduced memory footprint and computational load. The effectiveness and efficiency of the proposed solution have been successfully evaluated on a public-available benchmark for indoor presence detection and a real-world scenario for in-car presence detection.

Future works will encompass always-on scenarios for the proposed solutions, incremental learning mechanisms to support the on-device learning, the extension of the use of UWB-radar to human activity recognition and the use of adaptive mechanisms to support the on-device learning in presence of concept drift (e.g., faults affecting the device or changes in the environmental conditions in which the device is operating).

Acknowledgment

The authors would like to thank Ing. P. Lento, and Dr. A. Bassi from Trusense s.r.l. and Ing. G. Viscardi from Politecnco di Milano for the valuable support in the project.

References

[1] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[2] C. Alippi, *Intelligence for embedded systems*. Springer, 2014.

[3] C. Alippi, R. Fantacci, D. Marabissi, and M. Roveri, "A cloud to the ground: The new frontier of intelligent and autonomous networks of things," *IEEE Communications Mag.*, vol. 54, no. 12, pp. 14–20, 2016.

[4] C. Alippi and M. Roveri, "The (not) far-away path to smart cyber-physical systems: An information-centric framework," *Computer*, vol. 50, no. 4, pp. 38–47, 2017.

[5] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," *arXiv preprint arXiv:2103.13630*, 2021.

[6] J. Liu, S. Tripathi, U. Kurup, and M. Shah, "Pruning algorithms to accelerate convolutional neural networks for edge applications: A survey," *arXiv preprint arXiv:2005.04275*, 2020.

[7] S. Disabato and M. Roveri, "Reducing the computation load of convolutional neural networks through gate classification," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[8] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama, "Adaptive neural networks for efficient inference," in *International Conference on Machine Learning*. PMLR, 2017, pp. 527–536.

[9] V. J. Reddi and al., "Widening access to applied machine learning with tinyml," *arXiv preprint arXiv:2106.04008*, 2021.

[10] R. Sanchez-Iborra and A. F. Skarmeta, "Tinyml-enabled frugal smart objects: Challenges and opportunities," *IEEE Circuits and Systems Magazine*, vol. 20, no. 3, pp. 4–18, 2020.

[11] E. U. Council, "Regulation (eu) n2016/679, art. 4," 2016.

[12] J. D. Taylor, "Ultra-wideband radar overview," in *Introduction to ultra-wideband radar systems*. CRC Press, 2020, pp. 1–10.

[13] A. Dundar and al, "Embedded streaming deep neural networks accelerator with applications," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 28, no. 7, pp. 1572–1583, 2016.

[14] Y. E. Wang, G.-Y. Wei, and D. Brooks, "Benchmarking tpu, gpu, and cpu platforms for deep learning," *arXiv preprint arXiv:1907.10701*, 2019.

[15] R. David, J. Duke, and al., "Tensorflow lite micro: Embedded machine learning for tinyml systems," *Proceedings of Machine Learning and Systems*, vol. 3, pp. 800–811, 2021.

[16] I. Fedorov, R. P. Adams, M. Mattina, and P. Whatmough, "Sparse: Sparse architecture search for cnns on resource-constrained microcontrollers," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[17] A. Kumar, S. Goyal, and M. Varma, "Resource-efficient machine learning in 2 kb ram for the internet of things," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1935–1944.

[18] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[19] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1389–1397.

[20] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," in *Proc. IEEE Conf. on computer vision and pattern recognition*, 2017, pp. 5918–5926.

[21] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *International conference on machine learning*. PMLR, 2015, pp. 1737–1746.

[22] X. Lin and al., "Towards accurate binary convolutional neural network," *Advances in neural information processing systems*, vol. 30, 2017.

[23] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*. Springer, 2016, pp. 525–542.

[24] C. Alippi, S. Disabato, and M. Roveri, "Moving convolutional neural networks to embedded systems: the alexnet and vgg-16 case," in *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 2018, pp. 212–223.

[25] A. Chowdhery, P. Warden, J. Shlens, A. Howard, and R. Rhodes, "Visual wake words dataset," *arXiv preprint arXiv:1906.05721*, 2019.

[26] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[27] J.-E. Kim, J.-H. Choi, and K.-T. Kim, "Robust detection of presence of individuals in an indoor environment using ir-uwb radar," *IEEE Access*, vol. 8, pp. 108 133–108 147, 2020.

[28] S. Chang and al., "An algorithm for uwb radar-based human detection," in *2009 IEEE Radar Conference*. IEEE, 2009, pp. 1–6.

[29] Y. Kim and T. Moon, "Human Detection and Activity Classification Based on Micro-Doppler Signatures Using Deep Convolutional Neural Networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 1, pp. 8–12, Jan. 2016.

[30] A. R. Diewald and al, "RF-based child occupation detection in the vehicle interior," in *2016 17th International Radar Symposium (IRS)*, May 2016, pp. 1–4.

[31] A. Caddemi and E. Cardillo, *Automotive Anti-Abandon Systems: a Millimeter-Wave Radar Sensor for the Detection of Child Presence*, Oct. 2019, pages: 97.

[32] X. Li, Y. He, and X. Jing, "A Survey of Deep Learning-Based Human Activity Recognition in Radar," *Remote Sensing*, vol. 11, no. 9, p. 1068, Jan. 2019, number: 9.

[33] J. Park and al., "Micro-Doppler Based Classification of Human Aquatic Activities via Transfer Learning of Convolutional Neural Networks," *Sensors (Basel, Switzerland)*, vol. 16, no. 12, p. 1990, Nov. 2016.

[34] S. An, G. Bhat, S. Gumussoy, and U. Ogras, "Transfer Learning for Human Activity Recognition using Representational Analysis of Neural Networks," *arXiv:2012.04479 [cs, eess]*, Feb. 2021, arXiv: 2012.04479.

[35] Y. Lang, C. Hou, Y. Yang, D. Huang, and Y. He, *Convolutional neural network for human micro-Doppler classification*, Oct. 2017.

[36] V. C. Chen, "Analysis of radar micro-doppler with time-frequency transform," in *Proc. IEEE Workshop on Statistical Signal and Array Processing*. IEEE, 2000, pp. 463–466.

[37] V. C. Chen, D. Tahmoush, and W. J. Miceli, *Radar Micro-Doppler Signatures*. Institution of Engineering and Technology, 2014.

[38] M. Chmurski, M. Zubert, K. Bierzynski, and A. Santra, "Analysis of Edge-Optimized Deep Learning Classifiers for Radar-Based Gesture Recognition," *IEEE Access*, vol. 9, pp. 74 406–74 421, 2021.

[39] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[40] F. J. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978.

[41] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[42] B. Jacob, S. Kligys, and al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[43] Z. Zhengliang, Y. Degui, Z. Junchao, and T. Feng, "Dataset of human motion status using ir-uwb through-wall radar," *Journal of Systems Engineering and Electronics*, vol. 32, no. 5, pp. 1083–1096, 2021.

[44] A. G. Howard, M. Zhu, and al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.