TinyssimoRadar: In-Ear Hand Gesture Recognition with Ultra-Low Power mmWave Radars

Andrea Ronco
Center for Project-Based Learning
D-ITET, ETH Zürich
aronco@ethz.ch

Philipp Schilk
Center for Project-Based Learning
D-ITET, ETH Zürich
schilkp@ethz.ch

Michele Magno
Center for Project-Based Learning
D-ITET, ETH Zürich
magnom@ethz.ch

***Abstract*— Smart Internet of Things (IoT) devices are on the rise in popularity, with innovative use cases and applications emerging every year. Including intelligence in these novel systems presents the challenge of integrating interaction and communication in scenarios where traditional interfaces are not viable. Hand Gesture Recognition (HGR) has been proposed as an intuitive Human-Machine Interface, potentially suitable for controlling several classes of devices in the context of the Internet of Things. This paper proposes a low-power in-ear HGR system based on mm-wave radars, efficient spatial and temporal Convolutional Neural Networks and an energy-optimized hardware design. The design is suitable for battery-operated devices, with stringent size and energy constraints, enabling user interaction with wearable devices, but also suitable for home appliances and industrial applications. The proposed machine learning model is characterized thoroughly for robustness and generalization capabilities, achieving 94.9% (single subject) and 86.1% (Leave-One-Out Cross-validation) accuracy on a set of 11+1 gestures with a model size of only 36 KiB and inference latency of 32.4 ms on a 64 MHz Cortex-M33 microcontroller, making it compatible with real-time applications. The system is demonstrated in a fully integrated, miniaturized in-ear device with a full-system average power consumption of 18.4 mW, a more than 6x improvement on the current state of the art.**

***Index Terms*—mm-wave, radar, gesture recognition, low-power, embedded, sensor**

I. Introduction

The past decade has witnessed an extremely rapid expansion of technology, with intelligence today reaching domestic appliances, industrial machinery, and wearable systems [1]–[3]. Many of these devices are developed in the context of Internet of Things (IoT), and they exploit their connection to the internet to provide services and log data. With the expansion of intelligence to such a heterogeneous pool of IoT devices, the necessity of an appropriate interaction medium is gaining relevance [4]. Traditional interfaces, such as keyboards first and touchscreens later, offer a very rich and flexible interface, suitable for relatively complex devices. However, their limitations are becoming more evident when it comes to adapting to the evolving landscape of ubiquitous, and perhaps simple devices such as wearable systems, where the integration of traditional interfaces is not a viable option for physical constraints, strict energy requirements, or purely aesthetic reasons. The need for more intuitive and adaptive solutions has become increasingly evident, prompting extensive research into alternative technologies such as, but not limited to, voice interfaces [5], gaze estimation and eye-tracking [6], [7], and brain-computer interfaces based on Electroencephalogram (EEG) [8].

Hand Gesture Recognition (HGR) has emerged as a novel and promising alternative for Human-Machine Interface (HMI) [9] thanks to its natural and intuitive nature. Vision-based techniques for HGR have been extensively explored [10]–[12], thanks to the excellent state of the art in computer vision. Several setups based on RGB cameras have been proposed [13], with different approaches including color segmentation [14] and Convolutional Neural Network (CNN). While effective in certain scenarios, they often suffer from high sensitivity to the lighting conditions, and complexity of the background. Furthermore, they come with potential bias based on skin color [15] and with added privacy concerns. Methods based on depth cameras [16], [17] and RGB-D fusion [18], [19] have also been investigated, taking advantage of the depth information to aid the segmentation. However, vision methods usually require relatively expensive and bulky infrastructure and appropriate processing power, which makes them suitable in robotic systems or fixed installations, but hard to integrate into small, discrete IoT applications.

Other approaches for HGR focus on wearable systems, such as Surface Electromyography (sEMG) wristbands [20], which measure the electrical activity related to muscle activation and correlate it with hand gestures, typically using statistical approaches or deep learning [21], [22]. While promising, these solutions suffer from generalization issues when not tuned to a specific user [23]. Other solutions based on Inertial Measurement Units (IMUs) make use of sensing gloves [24] or multi-node wearable devices [25] to tackle the problem, often resulting in complex and highly obtrusive systems. Approaches to fuse IMU data with sEMG [26] or charge variation sensing [27] also proved successful. However, all such solutions require an external device to be worn, which makes them a less desirable option for the heterogeneous setting of IoT.

mm-Wave radar technology has already been proposed as an alternative solution [28], offering a unique set of advantages for hand gesture recognition. Due to its functional mechanism and sensing medium, it offers contactless sensing, and is agnostic to environmental conditions, such as lightning, humidity, and dust. In recent years, substantial progress has

979-8-3503-7025-6/24/$31.00 ©2024 IEEE
DOI 10.1109/IoTDI61053.2024.00021

been made in the development of mm-wave radar-based HGR systems [9], [29]. Furthermore, radar data is intrinsically complex, requiring substantial processing to extract meaningful insights. Efficient and resource-conscious processing methods are required to operate within the constraints imposed by the battery capacity, which necessarily restricts the available computational resources and memory budget. However, little research has been conducted on real-world implementation challenges, such as addressing the issues of integration, energy budget, and processing power requirements.

This paper bridges this gap by demonstrating a fully integrated HGR system based on a novel low-power 60 GHz Frequency Modulated Continuous Wave (FMCW) radar. We showcase the system in the challenging scenario of a battery-operated earbud, demonstrating its feasibility even in small, highly energy-constrained devices. This is achieved by adopting a holistic approach, starting from the algorithm design, down to the efficient hardware and software implementation. We introduce an efficient tiny machine learning model capable of running on a low-power ARM Cortex-M33 microcontroller in real time with low latency and high classification accuracy, and only requiring a few KB of memory. We further provide an extensive analysis of the trade-offs between gesture recognition performance and computation complexity, thereby offering a comprehensive perspective for future researchers and developers in this field.

We believe that mm-wave radar technology has the potential to enable robust, context-aware HGR and to revolutionize the field of wearable technology. By bridging the current disconnect between technology and application, we hope to pave the way for a new era of intuitive, hands-free interactions and seamless integration of IoT devices into our daily lives. With the publication of this paper, we also release the dataset, the algorithmic implementation, and the hardware platform to further accelerate research in this field.

II. Related Work

Radar-based gesture recognition has already been demonstrated with various sensing technologies [29]. In [30] the authors interpret the micro-Doppler signature of hand motion with a Continuous Wave (CW) radar system operating at 24 GHz and a deep CNN, showing an accuracy over 90% with 14 gestures for single-subject datasets, but low generalization capabilities. A similar setup was used by [31] with a system at 5.8 GHz, also investigating the effect of distance to the sensor on the accuracy.

The authors of [32] propose a system based on Ultra-Wide Band (UWB) radars for mid-air digit writing. The described setup is composed of three nodes, and Time of Arrival (TOA) is used with a trilateration algorithm for hand tracking and digit recognition. This solution is, however, only suitable for non-mobile applications, such as inside a room or vehicle, as it requires involved infrastructure with multiple nodes at known relative distances. A system based on a single-channel UWB system was proposed in [33], and multiple classification techniques have been analyzed, obtaining accuracies over 90%

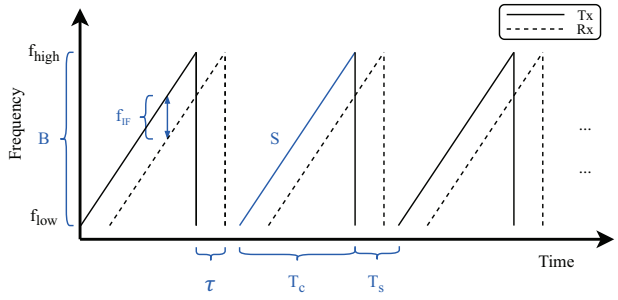

Fig. 1. Time-frequency diagram representing transmitted and received signals in FMCW radars. Marked in blue are some relevant parameters.

on a set of 14 gestures. A similar approach was used by [34] for in-vehicle finger counting. Employing a combination of UWB and inertial sensing, the authors of SeleCon [35] propose a HMI modality for IoT. However, this approach also requires UWB localization infastructure, as well as an UWB-enabled wristband or smartwatch.

FMCW radars have been proposed as an effective option for gesture recognition, offering simultaneously precise velocity measurements and direct range measurements, with the extra potential of zone segmentation for different gestures. A system based on a 60 GHz FMCW radar was demonstrated in [28], capable of classifying 4 dynamic gestures with 92.1% accuracy. The work was followed up by Wang et al. [36] with the development of a new machine learning architecture leveraging a CNN-based representation network operating in the Range-Doppler domain, combined with an Long Short-Term Memory (LSTM) network to model dynamic gestures. This work introduced a new set of 11 gestures, comprising some challenging classes that only involve fine motion of the fingers. A step towards a low-power implementation of the systems was accomplished in [37], where gesture recognition on the same gesture set was demonstrated running successfully on a Parallel Ultra-Low Power (PULP) chip with a 92 KiB model and estimated system-level power consumption of 120 mW.

III. Radar Background

Radars operate by emitting an electromagnetic signal and then measuring the characteristics of the reflections caused by the surrounding targets, which are received after a propagation delay, following an illuminate/echo mechanism. Radar taxonomy is mostly determined by the characteristics of the emitted signal, which also defines the type of information that can be measured. In the case of FMCW technology, the emitted signal is modulated in frequency, typically centered around a carrier frequency f_c. While there are various possible modulation schemes, such as triangular and sine wave modulation, this paper focuses on the widely used saw-tooth modulation, characterized by a linear frequency sweep from f_{min} to f_{max}. An illustration of this modulation is depicted in fig. 1, which also illustrates the signal's bandwidth, defined as the difference between f_{max} and f_{min}, as well as the modulation's slope S.

In the time domain, a chirp signal can be mathematically expressed as:

$$x_T(t) = A \sin\left(2\pi f_{min} t + \pi \frac{B}{T_c} t^2\right) \tag{1}$$

where A represents a generic magnitude term. A note worth mentioning is that the target is assumed to be static during the duration of a chirp T_c, which is often in the order of a few microseconds.

The echoed signal reflected by a target at distance d will have the same characteristics as the illumination signal with an extra time delay of $\tau = 2d/c$, as determined by the round-trip delay at the speed of light c. In FMCW radars the received echo signal is demodulated by mixing it with the transmitted signal, resulting in a low-frequency baseband representation called Intermediate Frequency (IF) signal. Assuming a target at distance d, the equation of the IF signal is

$$S_{IF}(t) = A \sin\left(2\pi \left(\frac{B}{T_c}\tau t + \frac{2d}{\lambda}\right)\right) \tag{2}$$

with λ being the wavelength of f_{min}, and A being a generic magnitude term that depends, among others, on antenna gain, distance of the target and its reflectivity. Equation 2 can be further simplified as:

$$S_{IF} = A \sin\left(2\pi f_{IF} t + \phi\right) \quad \text{where} \quad f_{IF} = \frac{B}{T_c}\frac{2d}{c} \tag{3}$$

which highlights the linear relationship between the target's distance d and the frequency of the IF signal f_{IF} enabling a target's distance to be resolved by analyzing the frequency content of the IF signal. This is typically achieved in discrete time using the Fast Fourier Transform (FFT) algorithm, an approach often referred to as the *Range FFT* in the literature. Each discrete frequency magnitude obtained using the FFT represents the reflected energy of targets within a fixed distance interval. This process discretizes the range measurements into multiple *Range Bins* with a resolution denoted as $d_{res} = \frac{c}{2B}$, which is solely dependent on the chirp bandwidth. The range resolution for commercial mm-wave radars typically falls within the centimeter range. In situations involving multiple targets at different distances, or targets that span across several range bins, the frequency contributions are simply superimposed. In practical scenarios, reflections from the environment and noise inevitably also contaminate the signal.

Simultaneously, it is possible to detect sub-millimeter *displacements* of the target within a range bin by measuring the phase change across different chirps separate by a known time T_s. FMCW radars exploit this effect to estimate the target's velocity by sending a fast train of chirps separated by a time interval T_s in what is called a radar *frame*. Within a frame, the phase in each range bin occupied by the target rotates at a speed proportional to the target's velocity, such that the velocity can again be extracted in the frequency domain with a second FFT along the chirps. From eqs. (2) and (3), given two chirps a small displacement Δd can be derived. By extension, the target's velocity v can be expressed as a function of the phase angular velocity, as shown in eq. (4).

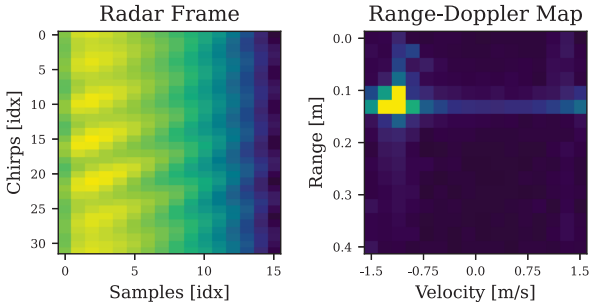

Fig. 2. Raw radar frame with 32 chirps and 16 samples (left), and the resulting Range-Doppler map (right).

$$\Delta d = \frac{\lambda\, \Delta\phi}{4\pi} \qquad v = \frac{\lambda \omega}{4\pi} \tag{4}$$

Combining these two ideas, a 2D frequency analysis can convert a radar frame into a distance-velocity map, referred to in the literature as *Range-Doppler* map. The map encodes the target's distance and velocity into a two-dimensional array, a sample of which is shown in fig. 2, alongside the respective raw radar frame.

IV. Dataset

The system was developed around the BGT60TR13D FMCW radar (Infineon Technologies), a novel sensor that targets low-power applications. The device operates at a frequency of 60 GHz, and is provided as an Antenna-in-Package chip which simplifies the integration effort. The sensor embeds one transmitting and three receiving antennas, physically placed in an L-shape.

A custom dataset had to be acquired, as no alternative with the same sensor was available. The dataset was acquired by replicating the gesture set proposed in [36]. The set encompasses 11 gestures of varying complexity, comprising both "micro-gestures" characterized by small movements of the fingers, such as *Pinch* gestures, *Finger Slide* and *Finger Rub*, and more dynamic gestures involving the full motion of the hand. Preserving the same set of gestures allows for a direct comparison with the existing work in [36] and [37]. The set was further expanded with one extra dummy gesture representing *Idle/No Activity* to allow meaningful inference when no hand is detected. This is an absolute necessity to extend the approach from an academic proof of concept to a fully integrated approach.

The dataset was acquired with a radar evaluation board and MATLAB SDK from the manufacturer, which provides APIs for the both configuration of the device and data logging. The acquisition setup was placed on the flat surface of a table, with the subject sitting next to it and performing the gestures as needed. The segmentation of the gesture was achieved by

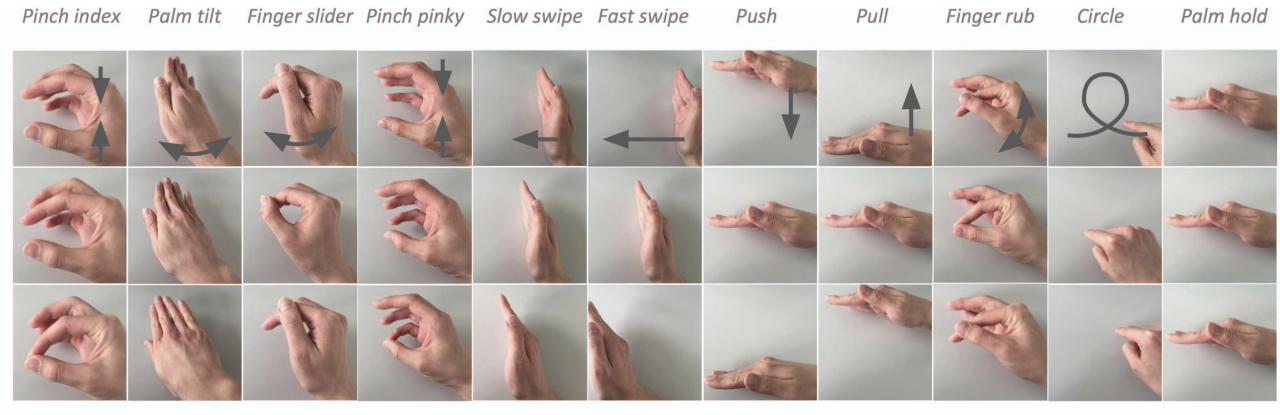

Fig. 3. Gesture set as defined in [36], replicated in our custom dataset. Figure also taken from [36].

the subject triggering each acquisition when ready to perform the gesture. The radar was configured with saw tooth chirp pattern from $58.5\,\mathrm{GHz}$ to $63.5\,\mathrm{GHz}$, for a total bandwith of $5\,\mathrm{GHz}$, and a chirp slope of $450\,\mathrm{MHz\,\mu s^{-1}}$. The individual chirp spacing was configured to be $700\,\mu\mathrm{s}$ for a sequence of 96 chirps, resulting in an overall framerate of $13.5\,\mathrm{FPS}$. Only 16 samples per chirp are acquired, which at $2\,\mathrm{MHz}$ sampling rate, and considering hardware overhead, gives a total chirp time of $13\,\mu\mathrm{s}$. The internal anti-alias filter was set to $600\,\mathrm{kHz}$, and the high-pass cutoff frequency was set to $180\,\mathrm{kHz}$ to reduce the effect of the DC leakage.

With this setup, two datasets were generated: a single-subject dataset (Subject 0) consisting of 700 samples per gesture, and a multi-subject dataset consisting of 50 samples per gesture from 20 subjects. The data from Subject 0 is used in both datasets. Prior to the recording session, the participants were provided with a demonstration of the gestures before being asked to replicate them. The gestures were performed $5\,\mathrm{cm}$ to $40\,\mathrm{cm}$ from the radar. This range was chosen as a consequence of the configured chirp slope and sampling time.

A. Dataset Augmentation

Only 32 chirps per frame are necessary for the algorithm pipeline to generate the Range-Doppler maps, as this already provides a sufficient velocity resolution of circa $0.2\,\mathrm{m\,s^{-1}}$. Nevertheless, the entire sequence of chirps is used for dataset augmentation: the training pipeline has been designed to generate the Range-Doppler maps on the fly extracting random crops of 32 chirps from the full sequence of 96 available in the dataset. This technique allows sampling frames from the original sequence with a variable time shift, with the beneficial effect of augmenting the variability of the training data.

V. Preprocessing

While deep learning has proved effective in feature extraction directly from raw data with large models [38], their complexity falls beyond the capabilities of an embedded platform. Instead, knowledge-driven pre-processing can greatly improve the performance of the model with low computation overhead by simplifying the task. Practically all previous research in the field of radar-based HGR relies on a pre-processing step in the frequency domain [17], [39]. The clear advantage of this approach has been shown with a comparative analysis in [37].

As introduced in section III, the range and velocity information of the targets are well represented in the Range-Doppler domain. Furthermore, the locality of information on velocity and range is better represented in this domain where they both map linearly. This is advantageous when working with CNNs due to the intrinsically local scope of the kernels. For these reasons, the radar frames are first pre-processed into Range-Doppler maps by the means of two FFTs, starting with a real FFT on the slow-time dimension (along the chirps), followed by a complex FFT on the samples axis (fast-time). The magnitude of the resulting complex image is computed and used as input for the neural network. We apply a further clipping to the Range-Doppler maps, bounding the values between 0 and 1 as a form of normalization of the inputs of the neural network, since most of the information was observed to be in this range. Observations on the data also showed that the DC components of the signal (range 0) were always saturated. Thus this row was dropped as it did not introduce any useful information. With 32 chirps and 16 samples per chirp, the pre-processing produces Range-Doppler maps with a resolution of 15x16.

VI. Neural Network Architecture

A gesture is defined as a sequence of 16 radar frames, corresponding to a time window of about $1.2\,\mathrm{s}$. Each radar frame is pre-processed, producing a 15x16 Range-Doppler. An intuitive representation of the inputs of the Neural Network is presented in fig. 4.

A supervised learning approach was adopted to solve the problem of gesture classification. The neural network architecture, inspired by [37], was designed from the beginning to target deployment on constrained devices, both in terms of parameter count and available computational power. In addition, the approach was designed to be efficient in a

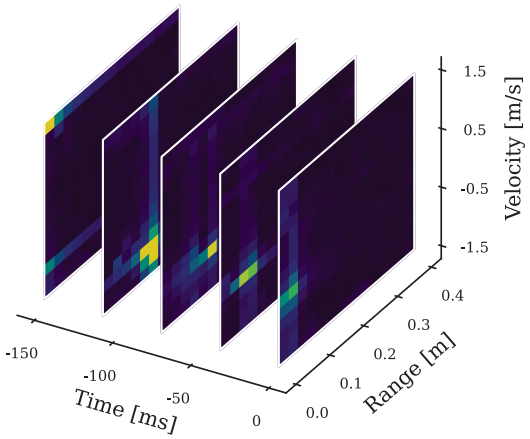

Fig. 4. Sequence of Range-Doppler maps associated with a gesture. A sequence of 16 frames composes the full input for the neural network.

streaming-oriented real-time setting, necessary in a real-world system.

The task is separated into a cascade of three sub-models. A convolutional Feature Extractor is used to extract spatial information from the Range-Doppler maps, which encode the instantaneous position and velocity pattern during the gesture. The second model exploits the abstract representations extracted by the first block, merging the temporal features spread across different frames into a single feature representation. Finally, a fully connected model is used to separate the gestures in the predefined set of classes defined at training time. A block diagram of the model is shown in fig. 5. In all layers, unless otherwise specified, ReLU activation was used for its efficient implementation.

A. Feature Extractor

While computing the Range-Doppler maps could be already seen as a feature extraction step itself, exploiting the spatial representation of range and velocity in this domain is beneficial. First, a convolutional Feature Extractor is used to extract features from the Range-Doppler maps, which encode the instantaneous position and velocity pattern during the gesture.

The first component of the Neural Network is based on a lightweight CNN composed of three 2D convolutional layers, with kernel sizes of 5x5, 3x3 and 3x3 respectively, and one pooling layer with max-pooling policy. The three layers used 16,24, and 16 channels respectively. Batch normalization [40] and dropout were also introduced to accelerate convergence and reduce overfitting. This network extracts spatial features from the Range-Doppler maps, producing a flat abstract representation. Another advantage of this approach is the reduction in runtime memory requirements, as the network compresses the inputs into a smaller set of features that can be stored more efficiently. The feature sets extracted by the CNN are stored in a shift buffer of length 16, which acts as input for the following step.

B. Temporal Convolutional Network

The second block is responsible for integrating the temporal information contained across several frames. Several architectures have been proposed in the last decade for temporal models. Recurrent Neural Network (RNN) such as LSTM and GRU are common choices for temporal sequences, as they allow modelling an infinitely long sequence by storing and updating an internal state, and they have been demonstrated on Edge devices [41]. However, these architectures have been criticized for having in fact a short memory, mostly due to vanishing gradient issues during training [42]. Furthermore, they tend to be memory and computationally intensive. Temporal Convolutional Networks (TCNs) have demonstrated to over-perform recurrent architectures in many fields [43]. They exploit a strategy of exponential kernel dilation, which allows them to extend the receptive field quickly while remaining relatively shallow. Causal convolutions are used in TCN architectures to model real-time applications, where only the past data points are known.

We use a TCN composed of 4 stacked convolutional layers with small kernels of size 2 and dilation from 1 to 8, which results in a receptive field of 16. All convolutions use 16 channels. In addition, skip connections between blocks and a dropout of 10% are used between the layers. An extra initial 1D convolutional layer with a unit kernel size and no activation function is used to adapt the dimensionality of the input space (number of spatial features) to the desired number of channels in the TCN. This block produces again a flat set of features, this time comprising both the spatial information and the temporal knowledge across several frames.

C. Classification Head

A simple Multi-Layer Perceptron (MLP) model is used to produce a prediction based on the features produced by the previous block. The classification head is composed of three fully connected layers with dropout. The two hidden layers use 32 units each, while the dimensionality of the output layer is 12, equal to the number of classes. ReLU activation is used in the hidden layers, and Softmax activation is used for the output layer, producing normalized class probabilities.

D. Training Policy

The model ensemble is trained with all components in the loop. During the training phase, the zero padding in the TCN exposes the network to partially complete gestures. This allows the network to reach high confidence levels as quickly as possible, even before the complete gesture has been seen. On the other hand, predictions based on the entire time window contain more information and should be considered more reliable. For this purpose, the custom Time-Weighted loss in eq. (5) was designed around the Categorical Cross-Entropy (CCE), in order to give more weight to predictions containing more frames.

$$\mathcal{L} = -\prod_{j=0}^{F-1} n(j) \cdot \sum_{i=1}^{C} y_{j,i} \cdot \log(p_{j,i}) \tag{5}$$

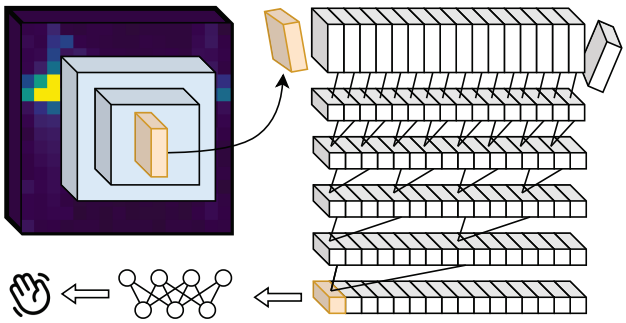

Fig. 5. Scheme representing the model described in section VI. The feature extractor is depicted on the top left. The feature set produced by the CNN is stored in the temporal shift buffer. The TCN (right) extracts temporal features from the shift buffer, used by the classification head (bottom left) for the final prediction.

$$n(x) = \tanh(\frac{x \cdot \pi}{F-1}) \tag{6}$$

Here, F and C are the number of frames and classes, $y_{j,i}$ and $p_{j,i}$ are the true and the predicted probability of the sample of frame j belonging to class i, and $n(j)$ is a monotonically increasing weight function that determines the weight based on how many frames were used to generate the prediction. In our training loop, eq. (6) was used as the weight function.

E. Quantization and Deployment

Since the deployment targets a real-time system, the padding step in the TCN is skipped, as the prediction is always based on the latest spatial features. This has the added benefit of reducing the run-time memory requirements and complexity of the model once deployed. Post training full integer quantization was chosen for the deployment on our custom embedded platform. Convolution kernels are quantized to single-byte signed integers (int8), while the biases retain a larger dynamic range with int32 quantization. Quantization significantly reduces the model footprint and allows the exploitation of the fixed-point hardware acceleration commonly available on embedded platforms. The model was deployed with TensorFlow Lite for Microcontrollers, which offers support for optimized kernels based on the efficient Digital Signal Processing (DSP) library CMSIS-NN [44] on Cortex-M-based platforms. The library makes use of efficient Single-Instruction-Multiple-Data (SIMD) instructions for Multiply-Accumulate (MAC) operations, granting a significant speed advantage over a pure software implementation.

VII. System Design

The selected algorithms are demonstrated on a system built on *VitalCore*, our custom hardware platform designed for low-power wearable sensor applications.

While a variety of applications could gain from the benefits provided by HGR as an interface, we choose to present the system within the demanding configuration of a wireless earbud. This underlines its integration capabilities and minimal energy requirements. Nonetheless, we want to emphasize that deploying this solution to a whole range of systems would be equally feasible - if not simpler, due to the removal of design constraints.

The system is composed of two main components. The *VitalCore* platform hosts the main application processor, power management, and all other non-application-specific hardware. The board is then expanded with a second Printed Circuit Board (PCB). referred to as *RadarPack*, which hosts the radar-related hardware. A custom 3D printed case, battery and power switch complete the design holding all components together. A high-level overview of the unit is shown in figure 6.

A. VitalCore

VitalCore is our custom research platform for wearable sensing technology. The device represents the evolution of *VitalPod* [45], and is designed around the two principles of energy efficiency and flexibility: the first a necessity for research in the wearable domain; the second a key advantage for rapidly exploring novel concepts. The board is a self-contained system that can be combined with extra application-specific hardware by means of so-called *VitalPack* extension boards, thanks to the 50-pin high-density connector on the bottom side. The platform can be seen on the top side in fig. 7, and was successfully validated in our previous work [46].

VitalCore is built around the ultra-low power nRF5340 System-on-Chip (SoC) (Nordic Semiconductor), which features both ample processing power through its dual-core ARM Cortex-M33 processors with a maximum clock frequency of 128MHz, and space through its built-in $1\,\mathrm{MiB}$ of flash and $512\,\mathrm{KiB}$ of RAM. It supports Bluetooth 5.4 standard and comes with a compact chip antenna with $2.4\,\mathrm{dBi}$ peak gain for an excellent wireless range. A full-speed USB interface provides flexible communication options for different application scenarios, debugging and data offloading.

The power management is controlled by a Power Management IC (PMIC) (MAX77654, Analog Devices), providing an efficient and flexible power subsystem. The integrated battery interface, monitor, and charger allow most re-chargeable batteries to be connected directly to *VitalCore*, while three software-controlled buck-boost converters with an output voltage range of $0.8\,\mathrm{V}$ to $5.5\,\mathrm{V}$, and two software-controlled

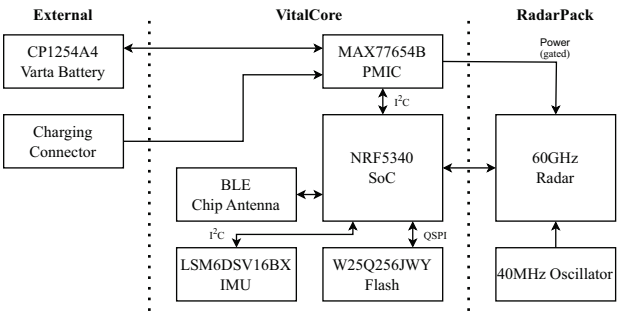

Fig. 6. Complete hardware system block-diagram of the custom earbud demonstration platform.

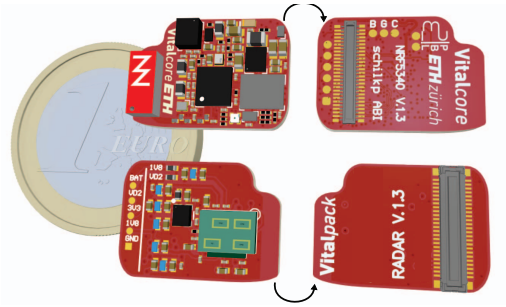

Fig. 7. Render of the custom *VitalCore* (top) and *RadarPack* (bottom) printed circuit boards, depicted next to a one euro coin for reference. The back side of both boards is also shown to present the high-density interface used to connect them.

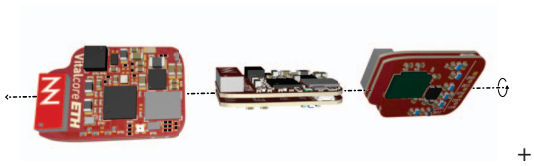

Fig. 8. Render of the custom *VitalCore* and *RadarPack* assembly, as installed in demonstration system.

Low-Dropout (LDO) linear regulators with an output range of $0.8\,\mathrm{V}$ to $3.975\,\mathrm{V}$ can accommodate the power-requirements of most extension applications. The SoC is powered by a dedicated high-efficiency buck converter (MAX38640, Analog Devices). *VitalCore* offers $256\,\mathrm{Mbit}$ of non-volatile flash memory (W25Q256JWY, Winbond) that can be used to store logging information, sensor data or other application-specific configuration. Finally, a multi-color LED with driver (IS31FL3194, Lumissil) and an IMU with *QVAR* fronted (LSM6DSV16BX, ST Microelectronics) provide an option for visual feedback and basic tap-based control.

The *VitalCore* has proven to be an effective platform for the rapid development of integrated devices in our previous work [removed for review]. With the publication of this work, we are open-sourcing the design, in the hope of encouraging more interesting applications.

B. RadarPack

The *RadarPack* is developed as an expansion board sharing the same form factor of the *VitalCore* to simplify the final assembly, as visible on the bottom of fig. 7. The board hosts the mm-Wave $60\,\mathrm{GHz}$ radar sensor BGT60TR13D, a $40\,\mathrm{MHz}$ reference oscillator required from the radar, and several passive filters for the power supply of the sensing element. The two power rails are directly derived from the PMIC from the main board, which allows to power-gate the entire board when needed. The two boards are coupled with the high-density connector, resulting in the assembly shown in fig. 8.

C. Mechanical Design and System Integration

The electronics are housed in a custom 3D printed enclosure shown in fig. 10, comparable in size and shape to commercial true-wireless earbuds that can be worn comfortably. A Lithium-Ion rechargeable battery with capacity $70\,\mathrm{mA\,h}$ (circa $260\,\mathrm{mW\,h}$) is used as the power source of the system and can be charged using a connector held in the stem of the casing. The device can be powered down completely by disconnecting the battery with a power switch, also placed in the stem. This is functional purely for development purposes, as the power subsystem of the device is capable of pseudo-shutdown mode, reducing the power consumption to a low level suitable for storage, and includes low-voltage battery protection. The complete assembly is shown in fig. 10.

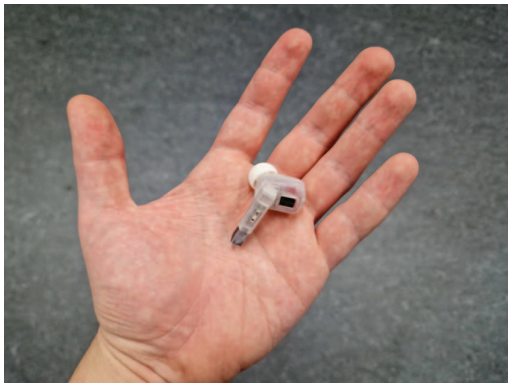

Fig. 9. Assembled device.

VIII. Results

Several experiments were conducted on the algorithms to evaluate the system performance and explore potential trade-offs to exploit. These results are reported in the subsection *Model Evaluation*. Results relative to the embedded system design and implementation follow, including power and latency evaluation.

A. Model Evaluation

Two metrics were used in our evaluation to assess the network's performance. Given a sequence of frames, the accuracy obtained on the last frame is the most intuitive metric to use since the last prediction in the sequence is expected to be the most accurate. However, we further evaluate the performance with a more robust metric by pooling the accuracy over the last four frames, which results in overall better predictions thanks to its smoothing effect. This metric was also used in [36], but across the entire sequence. While feasible, pooling on the entire sequence would add a considerable delay in the final prediction output on the real system. Since the network targets a constrained device, we further compare full-precision models with their quantized version.

The model is evaluated with both the single-subject and the multi-subject datasets. Leave-one-out cross-validation (LOOCV) validation was also introduced to assess the generalization capabilities of the gesture recognition system. During this experiment, the data from one subject was kept out of the training set and used instead as validation data. All the

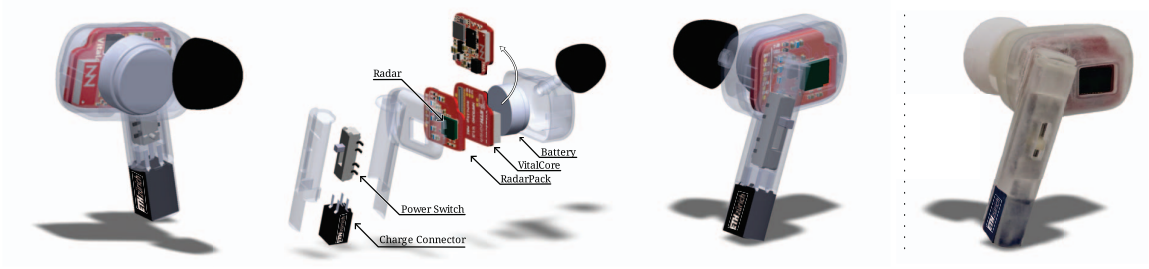

Fig. 10. 3D render of the system with individual components labeled, and picture of the assembled device.

	Single Subject	Multi Subject	LOOCV
Acc. (FP)	96.5% ± 1.6%	89.7% ± 1.5%	88.1% ± 3.7%
Acc. Pool. (FP)	97.0% ± 1.2%	90.2% ± 1.6%	88.3% ± 3.7%
Acc. (Q)	93.8% ± 3.2%	87.1% ± 4.5%	85.7% ± 5.4%
Acc. Pool. (Q)	94.9% ± 3.2%	87.9% ± 4.7%	86.1% ± 5.4%

TABLE I
MODEL EVALUATION. ACC. REFERS TO ACCURACY, POOL. REFERS TO FRAME POOLING OVER THE LAST 4 FRAMES, FP AND Q REFER TO FULL-PRECISION AND QUANTIZED MODELS.

	1 Ant.	2 Ant.	3 Ant.
1 Subj. Acc. Pool. (FP)	96.5%	96.4%	96.8%
1 Subj. Acc. Pool. (Q)	93.4%	91.8%	91.7%
LOOCV Acc. Pool. (FP)	88.5%	89.5%	87.6%
LOOCV Acc. Pool. (FP)	85.0%	85.9%	82%

TABLE II
ACCURACY FOR DIFFERENT NUMBERS OF RECEIVING ANTENNAS.

evaluations are repeated multiple times (40+ per subject) to assess the variability of the results and measure the capability of the network to produce consistent results over time. The mean accuracy and its standard deviation are reported in table I.

B. Per-Class Study

The confusion matrixes in fig. 11 show another interesting observation. Analysing the accuracy on a per-gesture basis indicates that a lot of errors happen in distinguishing between the two gestures *Pinch Index* and *Pinch Pinky*. Indeed the two gestures are similar, especially if translated in the Range-Doppler domain. By reducing the scope of the classification by a single gesture (through removal or merging of the two), the overall accuracy can be significantly increased. We conducted an evaluation without loading the *Pinch Pinky* gesture, which led to a 94.6% ± 4.6% LOOCV accuracy for the full precision model, and 90.9% ± 7% for the quantized one. Surprisingly, the effect of frame pooling in this case was not noticeable.

C. Per-Subject Study

The robustness of the methods was further examined with a per-subject analysis. This evaluation brings further insights into the performance of the LOOCV by showing the average performance of specific subjects with respect to each other. As visible in fig. 12, a mean accuracy over 80% is retained by virtually all subjects with LOOCV, even if a relevant drop can be seen in a few cases. This observation is relevant in the context of truly subject-agnostic systems where pre-training or fine-tuning is not an option, as it shows that there were no big outliers.

D. Number of Antennas

An ablation study on the number of antennas was conducted to explore the possibility of reducing the number of receiving antennas. In our implementation, the Range-Doppler maps generated with the data from the three antennas are directly used as input channels in the Feature Extractor block. While a reduction in the number of antennas would slightly decrease the memory and computation requirements, the greatest benefit would come from the integration viewpoint, as the sensor size could be reduced by at least 50%. The evaluation was conducted both with the single-subject dataset and with the multi-subject dataset with LOOCV, and the results are reported in table II. In the case of the multi-user analysis, no clear correlation could be identified, as the obtained accuracy was well within the previously evaluated standard deviation error. This suggests that the number of antennas does not have a significant effect, and applications could be based on a single-antenna system.

E. Complexity Trade-offs

Since the proposed architecture is composed of three components working together, a further evaluation was performed to explore the trade-off of network complexity versus accuracy. The computational load of the network is mostly concentrated in the Feature Extractor and in the TCN. We explored separately different configurations of the three blocks, and for each evaluated the mean accuracy and standard deviation with the LOOCV method. For the Feature Extractor and Classification Head, while a positive correlation between higher complexity and mean accuracy was identified, the gains were small and always below the standard deviation of the LOOCV. Clear gains were instead observed when tuning the complexity of the TCN by increasing the number of convolutional channels, as shown in fig. 13. It can also be observed that the effect is especially pronounced with the quantized version, due to the bottleneck effect caused by the TCN when too few channels are available. The evaluation shows a clear accuracy improvement by increasing the number of TCN channels up to 16,

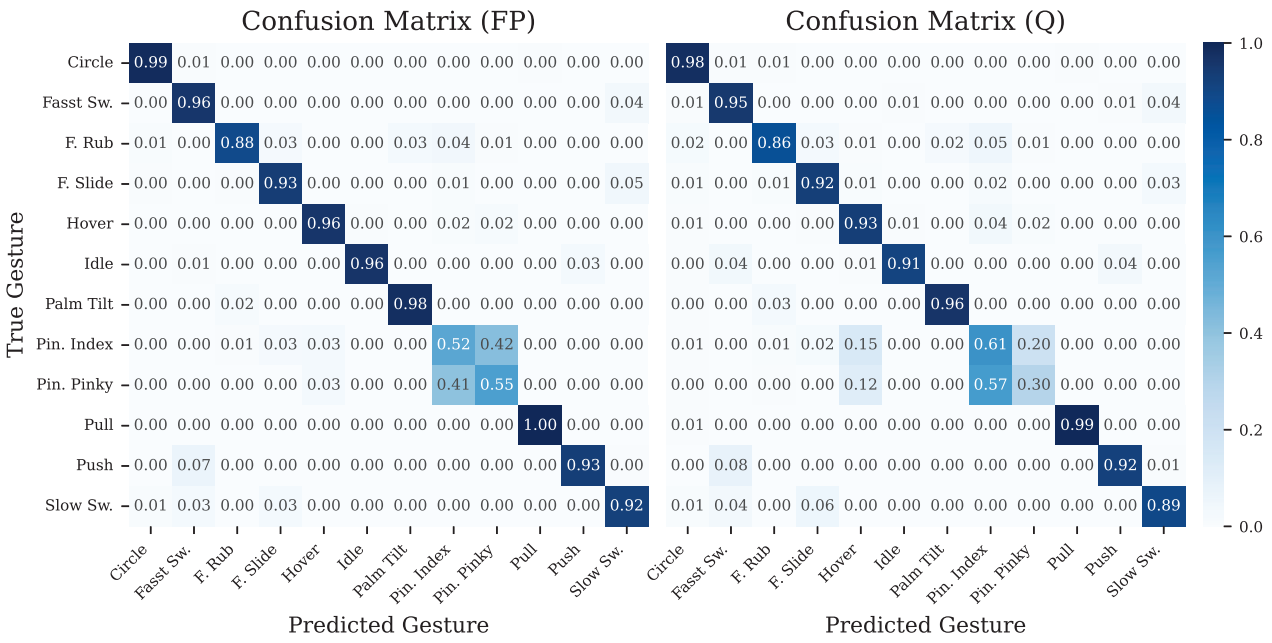

Fig. 11. Confusion Matrix for both full precision (left) and quantized model (right).

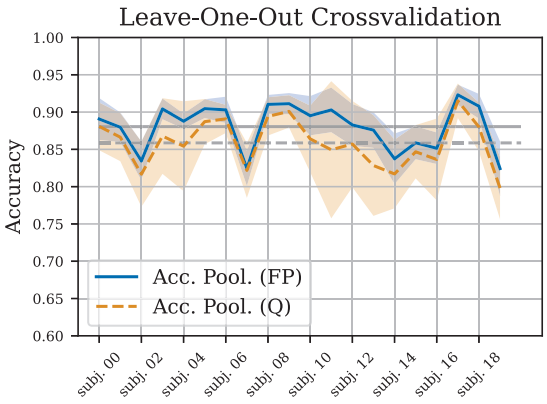

Fig. 12. Per-subject LOOCV mean accuracy over multiple train folds with the respective standard deviation as shaded fill. The average across all subjects is shown in grey.

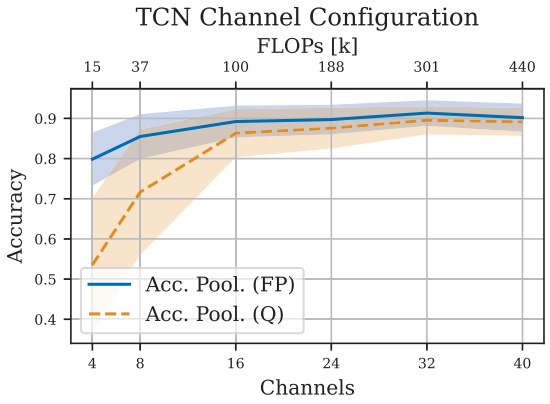

Fig. 13. Accuracy against number of channels in TCN. Diminishing returns are evident for more than 16 channels.

and diminishing returns with more channels. For this reason, 16 channels were used in the final embedded implementation, as it represents a good trade-off point in terms of computation complexity and parameter count.

F. Systems evaluation

Given the real-time requirements for embedded HGR, the processing latency is a critical metric, as the power consumption is critical in the context of battery-operated devices. In general, the processing must be faster than the time interval between new data samples. Ideally, an extra margin of time is desirable to allow further applications to execute on the same platform. We evaluate the latency of all the processing steps by running the inference on the platform and measuring the elapsed time with the internal RTC clock. The time is accumulated over 100 repetitions to reject the possible interference of context-switching or other sporadic high-priority tasks in the firmware, such as the battery monitor. The latency of the preprocessing step was consistently 3.69 ms. A detailed breakdown of the neural network processing latency, size, and complexity is reported in table III.

The power consumption of the system has been profiled both during idle time and during inference. For all the power

	CNN	TCN	Class.	Tot.
MAC	493,320	50,400	1988	545,708
Parameters	8,168	3,664	1,996	13,828
Time [ms]	23.53	4.89	0.27	32.39
Size [KiB]	13	18	5	36

TABLE III
BREAKDOWN OF LATENCY FOR ALL THE PROCESSING STEPS ON THE EMBEDDED PLATFORM. MEASURED AT CLOCK FREQUENCY 64 MHz.

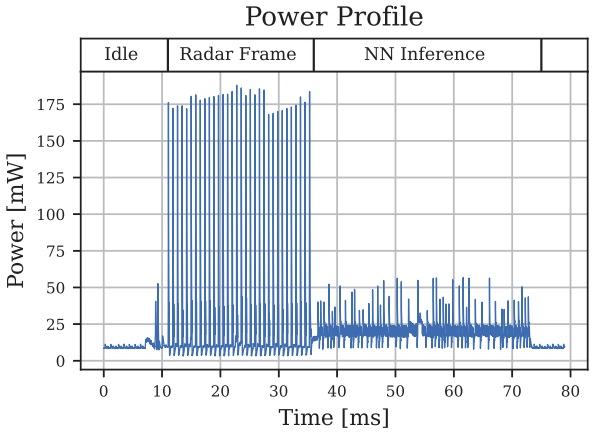

Fig. 14. Power profile during acquisition and processing of one radar frame. Also visible is a BLE event at time 10 ms.

and energy evaluations, a source at 3.7 V is assumed, simulating a standard lithium-ion battery. The average power consumption when the system is idle is circa 1.1 mW, measured with a Bluetooth Low-Energy (BLE) advertising interval of 100 ms. When the radar board is powered, a further 8.0 mW is used with most of the current required to power the 40 MHz oscillator. The power profile of one frame is visible in fig. 14. The radar activity is clearly visible as a sequence of 32 spikes in current, corresponding to the transmission of as many chirps. The average power consumption during the radar activity is 21.7 mW. The data processing step follows right after, with an average power consumption of 20.5 mW. Afterwards, the CPU sleeps until the next frame is available. This leaves around 41 ms of processing time for other tasks. Overall, the average power consumption of the entire system during HGR is 18.4 mW. With the battery included in our prototype (70 mA h, nominal voltage 3.7 V), the system could perform continuous gesture recognition for 14 hours. A battery of 120 mA h would be necessary to reach the 24-hour milestone.

IX. CONCLUSION

This paper presented a mm-wave-enabled earbud and an algorithm pipeline for Hand Gesture Recognition. A new efficient 3-stage machine learning model based on spatial and temporal convolutions is proposed as an optimized option for HGR in the Range-Doppler feature space. A thorough analysis of the model showed an accuracy of 97.0% (94.9% quantized) on a single-subject dataset, and 88.3% (86.1% quantized) LOOCV accuracy on 20 unseen subjects. The metrics could

	Wang et al. [36]	Scherer et al. [37]	This Work
S. Subj. Acc.	94.7%	92.4%	**94.9%**
M. Subj. Acc.	—	86.6%	**87.9%**
LOOCV Acc.	88.3%	78.9%	**86.1%**
Num. Subj.	10	**26**	20
Model Size	689 MiB	91 KiB	**36 KiB**
Sens. Power	300 mW	95 mW	**20.6 mW**
NN Power	—	21 mW	**11.4 mW**

TABLE IV
COMPARISON OF KEY PERFORMANCE METRICS WITH THE MOST RELEVANT WORK FROM WANG ET AL. [36] AND SCHERER ET AL. [37].

be improved even further by removing a single problematic gesture class which showed high confusion, boosting the LOOCV accuracy to 94.6% (90.9 quantized). Complexity trade-offs and the size of the antenna array were also explored, demonstrating the potential of a single-antenna system. With a final size of only 36 KiB, the model was deployed in a custom in-ear wearable device, demonstrating the first fully integrated wearable system for HGR based on a novel low-power FMCW radar sensor. The overall power consumption of the device is only 18.4 mW for continuous HGR. The on-device inference only takes 32.3 ms, sufficient for a real-time operation and to accommodate further tasks in the device.

A comparison of the most relevant metrics is shown in table IV. While we acknowledge that each work was developed with a different, custom dataset, maintaining the same gesture set allows for a direct comparison, showing that our method exceeds the previous work both in terms of accuracy and efficiency, with an improvement of more than 6x over the system-level power consumption.

Finally, we are happy to open source the model implementation[1] and the hardware platform *VitalCore*[2] on Github.

REFERENCES

[1] S. J. Ramson, S. Vishnu, and M. Shanmugam, "Applications of Internet of Things (IoT) – An Overview," in *2020 5th International Conference on Devices, Circuits and Systems (ICDCS)*, Mar. 2020, pp. 92–95.

[2] F. John Dian, R. Vahidnia, and A. Rahmati, "Wearables and the Internet of Things (IoT), Applications, Opportunities, and Challenges: A Survey," *IEEE Access*, vol. 8, pp. 69 200–69 211, 2020.

[3] M. Alaa, A. A. Zaidan, B. B. Zaidan, M. Talal, and M. L. M. Kiah, "A review of smart home applications based on Internet of Things," *Journal of Network and Computer Applications*, vol. 97, pp. 48–65, Nov. 2017.

[4] D. Mourtzis, J. Angelopoulos, and N. Panopoulos, "The Future of the Human–Machine Interface (HMI) in Society 5.0," *Future Internet*, vol. 15, no. 5, p. 162, May 2023.

[5] K. Seaborn, N. P. Miyake, P. Pennefather, and M. Otake-Matsuura, "Voice in Human–Agent Interaction: A Survey," *ACM Computing Surveys*, vol. 54, no. 4, pp. 81:1–81:43, May 2021.

[6] H. Admoni and B. Scassellati, "Social eye gaze in human-robot interaction: A review," *Journal of Human-Robot Interaction*, vol. 6, no. 1, pp. 25–63, May 2017.

[7] A. Bissoli, D. Lavino-Junior, M. Sime, L. Encarnação, and T. Bastos-Filho, "A Human–Machine Interface Based on Eye Tracking for Controlling and Monitoring a Smart Home Using the Internet of Things," *Sensors*, vol. 19, no. 4, p. 859, Jan. 2019.

[8] X. Gu, Z. Cao, A. Jolfaei, P. Xu, D. Wu, T.-P. Jung, and C.-T. Lin, "EEG-Based Brain-Computer Interfaces (BCIs): A Survey of Recent Studies on Signal Sensing Technologies and Computational Intelligence Approaches and Their Applications," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 18, no. 5, pp. 1645–1666, Sep. 2021.

[1] https://github.com/ETH-PBL/TinyssimoRadar
[2] https://github.com/ETH-PBL/VitalCore

[9] L. Guo, Z. Lu, and L. Yao, "Human-Machine Interaction Sensing Technology Based on Hand Gesture Recognition: A Review," *IEEE Transactions on Human-Machine Systems*, vol. 51, no. 4, pp. 300–309, Aug. 2021.

[10] Z. Xia, Q. Lei, Y. Yang, H. Zhang, Y. He, W. Wang, and M. Huang, "Vision-Based Hand Gesture Recognition for Human-Robot Collaboration: A Survey," in *2019 5th International Conference on Control, Automation and Robotics (ICCAR)*, Apr. 2019, pp. 198–205.

[11] H. Liu and L. Wang, "Gesture recognition for human-robot collaboration: A review," *International Journal of Industrial Ergonomics*, vol. 68, pp. 355–367, Nov. 2018.

[12] J. Suarez and R. R. Murphy, "Hand gesture recognition with depth images: A review," in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, Sep. 2012, pp. 411–417.

[13] D. Sarma and M. K. Bhuyan, "Methods, Databases and Recent Advancement of Vision-Based Hand Gesture Recognition for HCI Systems: A Review," *SN Computer Science*, vol. 2, no. 6, p. 436, Aug. 2021.

[14] Z. Hu and X. Zhu, "Gesture detection from RGB hand image using modified convolutional neural network," in *2019 2nd International Conference on Information Systems and Computer Aided Education (ICISCAE)*, Sep. 2019, pp. 143–146.

[15] A. Khalil, S. G. Ahmed, A. M. Khattak, and N. Al-Qirim, "Investigating Bias in Facial Analysis Systems: A Systematic Review," *IEEE Access*, vol. 8, pp. 130 751–130 761, 2020.

[16] C. Wang, X. H. Shi, L. W. Liu, and S. C. Chan, "A marker-less two-hand gesture recognition system using kinect depth camera," in *2015 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, Sep. 2015, pp. 1–4.

[17] T. Q. Vinh and N. T. Tri, "Hand gesture recognition based on depth image using kinect sensor," in *2015 2nd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS)*, Sep. 2015, pp. 34–39.

[18] Q. Feng, C. Yang, X. Wu, and Z. Li, "A smart TV interaction system based on hand gesture recognition by using RGB-D Sensor," in *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*, Dec. 2013, pp. 1319–1322.

[19] J. Liu, K. Furusawa, T. Tateyama, Y. Iwamoto, and Y.-W. Chen, "An Improved Hand Gesture Recognition with Two-Stage Convolution Neural Networks Using a Hand Color Image and its Pseudo-Depth Image," in *2019 IEEE International Conference on Image Processing (ICIP)*, Sep. 2019, pp. 375–379.

[20] Y. Zhang, Y. Chen, H. Yu, X. Yang, and W. Lu, "Learning Effective Spatial–Temporal Features for sEMG Armband-Based Gesture Recognition," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6979–6992, Aug. 2020.

[21] J. Qi, G. Jiang, G. Li, Y. Sun, and B. Tao, "Intelligent Human-Computer Interaction Based on Surface EMG Gesture Recognition," *IEEE Access*, vol. 7, pp. 61 378–61 387, 2019.

[22] Q. Li, Z. Luo, R. Qi, and J. Zheng, "DeepTPA-Net: A Deep Triple Attention Network for sEMG-Based Hand Gesture Recognition," *IEEE Access*, vol. 11, pp. 96 797–96 807, 2023.

[23] R. N. Khushaba, "Correlation Analysis of Electromyogram Signals for Multiuser Myoelectric Interfaces," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 4, pp. 745–755, Jul. 2014.

[24] A. Zanzarukiya, B. Jethwa, M. Panchasara, and R. Parekh, "Assistive Hand Gesture Glove for Hearing and Speech Impaired," in *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)*, Jun. 2020, pp. 837–841.

[25] P.-F. Xu, Z.-X. Liu, F. Li, and H.-P. Wang, "A Low-Cost Wearable Hand Gesture Detecting System Based on IMU and Convolutional Neural Network," in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, Nov. 2021, pp. 6999–7002.

[26] S. Jiang, B. Lv, X. Sheng, C. Zhang, H. Wang, and P. B. Shull, "Development of a real-time hand gesture recognition wristband based on sEMG and IMU sensing," in *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec. 2016, pp. 1256–1261.

[27] E. Reinschmidt, C. Vogt, and M. Magno, "Realtime Hand-Gesture Recognition Based on Novel Charge Variation Sensor and IMU," in *2022 IEEE Sensors*, Oct. 2022, pp. 1–4.

[28] J. Lien, N. Gillian, M. E. Karagozler, P. Amihood, C. Schwesig, E. Olson, H. Raja, and I. Poupyrev, "Soli: Ubiquitous gesture sensing with millimeter wave radar," *ACM Transactions on Graphics*, vol. 35, no. 4, pp. 142:1–142:19, Jul. 2016.

[29] S. Ahmed, K. D. Kallu, S. Ahmed, and S. H. Cho, "Hand Gestures Recognition Using Radar Sensors for Human-Computer-Interaction: A Review," *Remote Sensing*, vol. 13, no. 3, p. 527, Jan. 2021.

[30] S. Skaria, A. Al-Hourani, M. Lech, and R. J. Evans, "Hand-Gesture Recognition Using Two-Antenna Doppler Radar With Deep Convolutional Neural Networks," *IEEE Sensors Journal*, vol. 19, no. 8, pp. 3041–3048, Apr. 2019.

[31] Y. Kim and B. Toomajian, "Hand Gesture Recognition Using Micro-Doppler Signatures With Convolutional Neural Network," *IEEE Access*, vol. 4, pp. 7125–7130, 2016.

[32] S. K. Leem, F. Khan, and S. H. Cho, "Detecting Mid-Air Gestures for Digit Writing With Radio Sensors and a CNN," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 4, pp. 1066–1081, Apr. 2020.

[33] S. Skaria, A. Al-Hourani, and R. J. Evans, "Deep-Learning Methods for Hand-Gesture Recognition Using Ultra-Wideband Radar," *IEEE Access*, vol. 8, pp. 203 580–203 590, 2020.

[34] S. Ahmed, F. Khan, A. Ghaffar, F. Hussain, and S. H. Cho, "Finger-Counting-Based Gesture Recognition within Cars Using Impulse Radar with Convolutional Neural Network," *Sensors*, vol. 19, no. 6, p. 1429, Jan. 2019.

[35] A. Alanwar, M. Alzantot, B.-J. Ho, P. Martin, and M. Srivastava, "SeleCon: Scalable IoT Device Selection and Control Using Hand Gestures," in *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*, Apr. 2017, pp. 47–58.

[36] S. Wang, J. Song, J. Lien, I. Poupyrev, and O. Hilliges, "Interacting with Soli: Exploring Fine-Grained Dynamic Gesture Recognition in the Radio-Frequency Spectrum," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, ser. UIST '16. New York, NY, USA: Association for Computing Machinery, Oct. 2016, pp. 851–860.

[37] M. Scherer, M. Magno, J. Erb, P. Mayer, M. Eggimann, and L. Benini, "TinyRadarNN: Combining Spatial and Temporal Convolutional Neural Networks for Embedded Gesture Recognition with Short Range Radars," Mar. 2021.

[38] S. Dara and P. Tumma, "Feature Extraction By Using Deep Learning: A Survey," in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, Mar. 2018, pp. 1795–1801.

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

[40] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. Lille, France: JMLR.org, Jul. 2015, pp. 448–456.

[41] V. S. Lalapura, J. Amudha, and H. S. Satheesh, "Recurrent Neural Networks for Edge Intelligence: A Survey," *ACM Computing Surveys*, vol. 54, no. 4, pp. 91:1–91:38, May 2021.

[42] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[43] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," Apr. 2018.

[44] L. Lai, N. Suda, and V. Chandra, "CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs," Jan. 2018.

[45] P. Schilk, K. Dheman, and M. Magno, "VitalPod: A Low Power In-Ear Vital Parameter Monitoring System," in *2022 18th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. Thessaloniki, Greece: IEEE, Oct. 2022, pp. 94–99.

[46] P. Schilk, N. Polvani, A. Ronco, M. Cernak, and M. Magno, "In-Ear-Voice: Towards Milli-Watt Audio Enhancement With Bone-Conduction Microphones for In-Ear Sensing Platforms," in *Proceedings of the 8th ACM/IEEE Conference on Internet of Things Design and Implementation*, ser. IoTDI '23. New York, NY, USA: Association for Computing Machinery, May 2023, pp. 1–12.