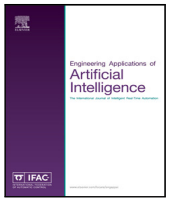
Engineering Applications of Artificial Intelligence 121 (2023) 106035

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Do we need early exit networks in human activity recognition?

Emanuele Lattanzi *, Chiara Contoli, Valerio Freschi

Department of Pure and Applied Sciences, University of Urbino, Piazza della Repubblica 13, Urbino, 61029, Italy

ARTICLE INFO

Keywords:
Deep learning
Early exit
Human activity recognition

ABSTRACT

Deep learning is nowadays considered state-of-the-art technology in many applications thanks to huge performance capabilities. However, the accuracy levels that can be obtained with these models entail computationally demanding resources. This results in a challenging task when such systems have to be deployed on edge devices with tight computing, memory, and communication requirements and when energy expenditure and inference delays have to be kept under control. Early exit is a design methodology aimed at reducing the burden of neural networks on computational resources, trading off accuracy for latency.

In this work, we aim at exploring the use of early exit for human activity recognition tasks. In particular, we propose an experimental assessment of the accuracy–latency trade-off on different deep network architectures across various publicly available datasets. We also evaluate the impact of early exiting in distributed environments by taking into account communication technologies.

Experimental results provide evidence of the significant gain provided by early exits in terms of latency (up to 35×), without a reduction in accuracy (in most cases), confirming the viability of an adaptive approach. In a distributed environment, early exit results are not beneficial in all situations. In particular, it is not convenient for models that are very fast (with inference latency lower than, or as equal as, that of communication) and for models that are forced to make extensive use of far exit points to satisfy the accuracy requirements. Therefore, communication delays in a distributed environment shape performance in an architecture-dependent way.

1. Introduction

Deep Learning has recently gained wide diffusion in many application fields, such as computer vision, natural language processing, healthcare monitoring or diagnostics, and activity recognition (Goodfellow et al., 2016; LeCun et al., 2015; Gu et al., 2021).

The adoption of this learning paradigm has reached super-human accuracy performance in many tasks. This has been made possible by the increasing availability of: *(i)* more powerful and specialized hardware (like graphical processing units, *GPU*, or specialized chips designed for acceleration purposes); *(ii)* curated datasets comprising thousands of examples, crucial for enabling the training of deep neural networks (DNN) models in supervised learning applications.

However, huge improvements in terms of accuracy have been achieved with more demanding models, from the computational requirements point of view. Indeed, adding layers and neurons, thus giving rise to deeper and wider models, unavoidably increases memory and compute needs, which is particularly problematic for the deployment on mobile devices and embedded systems, often characterized by strict memory, computational, communication, and energy constraints. Moreover, the number of operations needed by these types of neural networks to perform inference results in increased delays that could potentially prevent their adoption in some real-time scenarios. There is therefore a strong need for novel solutions that could circumvent memory and compute bottlenecks, through the design of systems that can, at the same time, minimize network size and guarantee fast inference without significant impact on the accuracy levels (Fedorov et al., 2019; Banbury et al., 2021; Lattanzi et al., 2022).

Several methodologies have been developed in this context to reduce the burden of networks size and its impact on system performance. Needless to say, all these approaches inherently trade-off improvements in memory usage and latency for accuracy. From the architectural point of view, for example, novel neural architecture search (NAS, for short) techniques allow more efficient exploration of the huge design space of possible network types and topologies, while sparsification and pruning techniques could lower the memory footprint through the reduction of the number of weights and of the synaptic links; other lines of research aim at implementing NN with reduced precision arithmetic.

Recent approaches have tackled the problem by means of the so called *early-exit* techniques. Early-exit deep neural networks (hereafter also denoted as EE-DNN), by mimicking a mechanism that is believed operational in humans neurobiology, adaptively trade-off accuracy for latency by exploiting the fact that not all input samples need the same amount of processing (i.e. they do not present the same difficulty level) to be analyzed. Hence, this type of models is composed of a backbone

* Corresponding author.
E-mail address: emanuele.lattanzi@uniurb.it (E. Lattanzi).

https://doi.org/10.1016/j.engappai.2023.106035
Received 17 June 2022; Received in revised form 2 February 2023; Accepted 20 February 2023
Available online xxxx
0952-1976/© 2023 Elsevier Ltd. All rights reserved.

network with additional branching exits (also known as exit points) that depart from the backbone along its length. Exit points act as intermediate classifiers that are progressively queried during inference. In this way, when a given sample is processed along the backbone network, intermediate classifications can be performed. If, according to a predefined criterion, a specific exit condition (or policy) is met, the prediction computed from the corresponding exit is taken as output. Otherwise the computation is forwarded to subsequent steps.

The main feature of EE-DNN solutions is represented by their capability of dynamically adapting to different loads (easy-to-process queries have high network exiting probabilities) and to different requirements of the underlying hardware platform (the same network could in principle be split across different distributed devices).

In recent years we have also witnessed continuous advancements in the design, development, and industrialization of products as, for example, smartphones or wearable devices, equipped with a wide range of sensors that represent key enablers for large scale, fine-grained monitoring of physical behavior (Perez-Pozuelo et al., 2021; Lattanzi and Freschi, 2020). Human activity recognition (hereafter also denoted as HAR) concerns the classification of the physical activities performed by human subjects from the signals extracted from sensor devices. For instance, HAR based on wearable/mobile sensing, aims to discriminate between different activities, like running, walking, standing, sitting, etc. This can be achieved through a workflow that is usually designed as: *(i)* window segmentation (e.g. sliding window partitioning); *(ii)* feature extraction and engineering; *(iii)* training of supervised learning models.

Early exit (hereafter also denoted as EE) has been, up to today, mainly applied to computer vision problems, which represent one of the most successful application domains of deep learning. Conversely, to the best of our knowledge, the adoption of early exit strategies for deep learning models applied to HAR has not been the subject of investigation, yet. In this work we aim at bridging this gap by making the following contributions:

1. We propose an empirical investigation of EE-DNN in human activity recognition tasks, across four different neural networks architectures, namely one-dimensional and two-dimensional convolutional networks, long short term memory networks (LSTM), and a mixed CNN–LSTM architecture. The effectiveness of the proposed models has been evaluated on four publicly available datasets, which have different characteristics in terms of data collection protocols and sensing platforms, to confer robustness to experimental results.
2. We carefully characterize, for the architectures under study, the trade-off between accuracy and latency during inference, by properly varying a threshold that controls the exit policy. The main findings of this empirical assessment are that EE does not negatively affect accuracy (with the exception of the two-dimensional CNN), while it enables significant speedup (up to 35× in terms of inference time).
3. We evaluate the effect of distributing EE-DNN models along four different devices connected by means of different communication technologies. In this case, differently from the previous setting, communication delays clearly affect the overall system performance, showing that EE is no longer always advantageous for some DNN architectures (namely LSTM and two-dimensional CNN).

The remainder of the article is organized as follows: in Section 3 we summarize the main state-of-the-art research in deep learning and in early exit techniques applied to DNN architecture design; in Section 4 we introduce the proposed approach, we describe the adopted EE networks and datasets, and we illustrate training and inference procedures; in Section 5 a description of the experimental setup used for evaluation of the system is provided (together with the related performance metrics); in Section 6 we report and discuss experimental results; in Section 7 we end by recapitulating the main contributions, making conclusive remarks and indicating possible future developments.

2. Related work

In recent years, human activity recognition has risen the interest of both academia and industry because of its high demand in different fields of applications. The literature is indeed rich in works that deal with HAR, spanning different research directions, both in terms of data source and focus of the investigation. In HAR, data might be generated from video or live cameras, mobile phones, and wearable sensors. Instead, in terms of investigation, topics are related to, for example: novel model proposal, existing model improvement/optimization, performance, and feature extraction improvement to cite a few. In this section, we review related work in the application of deep learning and EE techniques to HAR.

2.1. Hybrid deep learning for HAR

Automatic feature extraction, and more accurate and effective classification performance brought by deep learning methods pushed the research community to adopt those latter in the HAR context. In particular, two types of deep architectures, combined together, seem promising solutions for HAR classification problems, that is: convolutional neural networks (CNNs) and Long Short Term Memory (LSTM) networks.

Convolutional neural networks (CNNs) are a deep learning approach that recently became the state-of-the-art performance on several HAR datasets. The adoption of CNNs as reference architecture for HAR tasks coupled with HAR based on wearable/mobile sensing brings both benefits and challenges. Deep learning brings indeed to HAR two beneficial aspects: first, it avoids the need for manual feature design and extraction given its feature learning capabilities; second, it provides more accurate and, effective performance with respect to systems based on hand-crafted features, especially for the recognition of more complex activities (Hammerla et al., 2016; Wang et al., 2019). Performance improvement is however paid in terms of increased model complexity, that increased computational cost and computational resources that makes harder the deployment process of deep models on wearable and mobile devices. The most recent literature on CNNs applied to HAR proposes several approaches to tackle the challenge of improving the trade-off between recognition accuracy and resource consumption. To cite a few, Tang et al. propose the adoption of a novel CNN to learn multi-scale features representation (Tang et al., 2022). In particular, the authors proposed a hierarchical-split convolutional block in a convolutional network, which can capture a wide range of receptive fields within each feature layer. The goal was to learn richer feature representations to achieve higher accuracy while keeping model complexity low. In line with the purpose of serving real-time HAR applications on mobile and wearable devices, Cheng et al. proposed a new CNN adopting conditional computation (Cheng et al., 2022). This envisages replacing the traditional convolution with conditional parametrized convolution, which increases model performance without sacrificing inference performance. Channel-equalization is another novel lightweight CNN proposed by Huang et al. (2022) that aims at improving the ability of the network to learn useful feature representation, thus increasing the contribution to the activity recognition.

Back in 2016, Ordóñez and Roggen (2016) proposed DeepConvLSTM framework for wearable activity recognition which envisages convolutional and recurrent layers. Performance evaluation was carried out on the OPPORTUNITY dataset (Roggen et al., 2010), and proved to outperform pure feedforward neural networks by 4% on average on everyday activities recognition, and by 9% on average in an 18-class gesture recognition task. Deep and Zheng (2019) proposed to stack two 1D convolutional layers and one LSTM layer for activity recognition tasks. They passed smartphone data collected from the accelerometer and the gyroscope, specifically the UCI-HAR dataset (Reyes-Ortiz et al., 2016), through the CNN layers first, and the output is then passed as input to the LSTM layers to predict activities. The goal was to

prove that the proposed architecture outperformed pure LSTM and bidirectional LSTM networks given the same dataset. Mekruksavanich and Jitpattanakul (2021) proposed a generic HAR framework for smartphone sensor data, where four baseline LSTM networks are compared to study and analyze the impact of using different kinds of smartphone data from UCI-HAR, as well. A hybrid 4-layer CNN–LSTM network was also proposed to improve recognition performance in terms of accuracy. Similarly, Xia et al. (2020) proposed an architecture where data collected by mobile sensors are fed into two-layer LSTM followed by convolutional layers. They also applied global average pooling layer and batch normalization immediately after in order to reduce model parameters and speed up model convergence, respectively. Performance was then evaluated on three public dataset, proving a F1 score of 95.78% (UCI-HAR), 95.85% (WISDM Kwapisz et al., 2011) and 92.63% (OPPORTUNITY). A slightly different hybrid framework, AttSense, was proposed by Ma et al. (2019): AttSense is a multimodal neural network model that combines an attention mechanism via a CNN, with Gated Recurrent Units (GRU) network to capture both spatial and temporal domains from sensing signals. The framework was evaluated on three public dataset, providing a F1 score of 96.50% (Heterogeneous Stisen et al., 2015), 93.10% (Skoda Zappi et al., 2008) and 89.30% (PAMAP2 Reiss and Stricker, 2012). Another approach that leverages the combination of GRU and inception neural network by using various kernel-based convolutional layers is InnoHAR (Xu et al., 2019), which also showed good performance.

The reason behind such a successful approach is that the combined adoption of CNN and LSTM architecture allows for capturing both spatial and temporal features, which are two characteristics that are typical of human activities extracted from sensor devices. Compared to these works, our work aims at considering these state-of-the-art performance CNN and LSTM networks for HAR and carrying out a thorough exploration of the application of the early exit strategy methodology. Indeed, for deep learning models applied to HAR, early exit has not been the subject of investigation, yet. In particular, we propose an experimental assessment of the accuracy–latency trade-off on CNN and LSTM across various publicly available datasets. We also evaluate the impact of early exiting in distributed environments by taking into account communication technologies.

2.2. Early exit

The early exit (EE) approach aims at addressing higher latency and computational costs (introduced by deeper and larger networks at inference time) by implementing intermediate classifiers. Further details about this technique are provided in Section 3.

EE is well investigated in literature both in its local and distributed versions. Just to mention a few, recent literature discusses the design methodology of early exit networks highlighting its key components and recent advances in each of them (Laskaridis et al., 2021). Bonato and Bouganis (2021) proposed a design methodology for designing early exit networks by adding additional exit to a given CNN model that acts as a baseline. The goal was to reduce computation time compared to regular models, while maintaining, if not increasing, accuracy. Tan et al. (2021) focused instead on the threshold determination problem: typically, the trade-off between accuracy and latency is controlled by thresholds; therefore, in this work, Tan et al. investigated a latency-aware threshold optimization problem, where the goal was to maximize the overall inference accuracy while meeting the average latency requirement. Matsubara et al. (2021) surveyed the state-of-the-art in EE strategies by presenting a comparison of the most relevant approaches. It is worth mentioning that the survey highlights the fact that the research community mainly focused on EE strategies applied to Computer Vision and Natural Language Processing applications.

The other research direction is focused on distributed EE; in this case, the exit point is meant to be the (vertical) architectural layer, that is, the end-device, the edge, the fog, or the cloud layer. Here, the idea is that inference is carried out at the lowest level (i.e., the end-device), unless the result does not satisfy requirements, thus requiring further and more powerful capacity provided by higher levels. Teerapittayanon et al. (2017) proposed a distributed deep neural network that spans across distributed computing hierarchy, which allows fast and localized inference leveraging shallow portions of the neural network at the edge and end devices. Indeed, sections of the deep neural network are mapped onto the distributed computing hierarchy; by locating a few neural network layers on end devices or the edge, and many neural network layers in the cloud, they demonstrated that the network is able to scale vertically, and horizontally across multiple end devices, as well. Zhao et al. (2018) proposed the DeepThings framework for tightly resource constrained IoT edge clusters, whose goal was to adaptively distribute execution of CNN-based inference applications. The authors mainly focused on the distribution of early convolutional layers. In Li et al. (2019), a prototype named Edgent is proposed to leverage edge computing for deep neural network collaborative inference via deep neural network right-sizing, that is, via early exit inference at an appropriate intermediate deep neural network layer in order to accelerate inference, thus allowing a latency–accuracy trade-off. Similarly to Edgent, Laskaridis et al. (2020) proposed SPINN, a distributed progressive inference system equipped with method of selectively partitioning CNN execution across device and server setups, while also tuning the early exit policy. Last, Pacheco et al. (2021) recently demonstrated that early exit deep neural networks can classify most samples at the edge, thus avoiding sending data to the cloud, and reducing inference time.

Despite the rich literature related to EE, the adoption of early exit strategies for deep learning models applied to HAR has not been the subject of investigation, yet. To cite a recent attempt in line with this research direction, despite EE is not being explicitly mentioned, in the work proposed by Samie et al. (2020), authors partition human activity classes into two subsets; a hierarchical classification approach that decomposes the problem into three classifiers into two hierarchy layers is then proposed. The idea is that a partition can be classified using a light and accurate classifier directly on the IoT device. The other partition, since it might require a more complex and heavy classifier, shall be offloaded to a gateway. In Odema et al. (2021), EExNAS, that is, a design methodology to render one dimensional CNN-based wearable device solutions, is proposed for designing high-performance and resource-efficient dynamic Neural Architecture Search. Here, HAR is used as a use case study to prove the effectiveness of the proposed methodology on w-HAR dataset (Bhat et al., 2020). Results show that their model incurs a 0.584% drop in accuracy, but gains 78.985% and 47.076% in memory and energy efficiency, respectively. Rashid et al. (2022) proposed an adaptive CNN for energy-efficient HAR that makes the early exit decision based on output block predictor (instead of classification confidence as in traditional EE architecture).

Unlike these works, our work provides insights about the convenience of adopting EE-DNN architectures in human activity recognition tasks. We design and compare four different neural network architectures, which we then evaluate on four publicly available datasets, those latter different in terms of data collection protocols and sensing platforms. Each architecture is carefully characterized in terms of the trade-off between accuracy and latency during inference. Besides evaluating the local (i.e. on the same device) application of EE-DNN strategy, we also evaluate the impact of distributing EE-DNN models in a distributed environment (i.e. along four different devices connected by means of different communication technologies). To the best of our knowledge, none of the existing literature carries out such a thorough investigation.

3. Background

This section will present a background of deep neural networks and early exit networks. We begin with a quick introduction to two of the most widely adopted deep neural network models, followed by a brief

overview of early exit networks approach. The background includes basic principles and structure for neural networks, whereas most recent training, inference strategies, and metrics in order to properly design an early exit network are considered.

3.1. Deep neural networks

The concept of deep neural network (DNN) originates from a classical neural network: the deepness is given by the fact that multiple levels of a neural network can be placed between the input and the output layers. Such evolution allows for improved accuracy and performance. One of the most popular DNN is the convolutional neural network (CNN), which proved to be highly successful, especially in the field of image recognition and classification.

As detailed in Alzubaidi et al. (2021), a CNN architecture is typically composed of: *(i)* a convolutional layer, *(ii)* a pooling layer, *(iii)* an activation function, *(iv)* a fully connected layer, and *(v)* a classification layer. The convolutional layer takes as input a multi-channeled image, in n-dimensional format, and outputs a feature map. The pooling layer is in charge of keeping meaningful features while decreasing the size of the feature map. The activation layer allows us to learn other things, and its main role is to map the input to the output; the most well-known activation functions are: *Sigmoid*, *Tanh*, and *ReLU*, to cite a few. The fully connected layer acts as a classifier, and is normally placed at the end of the network, typically after the (last) pooling layer; the peculiarity of the fully connected layer is that each neuron is connected to all neurons of the predecessor layer. The final classification is however carried out by the classification layer via the loss function, which is responsible to evaluate the error, i.e., the difference between the predicted output and the actual one. The most commonly adopted function is *Softmax*, which provides a probability distribution of possible outcomes by converting a vector of real numbers given as input to the function.

It is worth noting that normally two-dimensional (2D) convolutional filters are used by CNN models to process 2D images; however, one-dimensional (1D) convolutional filters can be used by CNN models to process signals, and therefore can be employed to perform Human Activity Recognition (HAR). The opportunity to convert time series signals data to images is gaining a lot of momentum because it allows us to apply computer vision techniques and perform classification tasks. In HAR, this means that signals gathered from the triaxial accelerometer and gyroscope can be properly re-coded to images so that a "visual" analysis can be carried out to recognize, learn and classify patterns. Several re-coding techniques exist (Wang and Oates, 2015; Baldini et al., 2017; Qin et al., 2020), and in this paper we leverage the deep learning approach proposed by Wang and Oates (2015), in which time series are converted to Gramian Angular Summation/Difference Fields (GASF/GADF) images. Such an approach envisages the representation of time series as a polar coordinate system that produces one image for the GASF and the other for the GADF. The main advantage of such an approach is that temporal and spatial relations are preserved. Therefore, given an ad-hoc dataset, this preprocessing phase generates six images from the accelerometer and six from the gyroscopic data, leading to a total of 12 square images. An example of such an application is described in Section 4.

Another deep learning approach used in the HAR field is the Long Short Term Memory (LSTM) model, which is a type of a recurrent neural network system. As detailed in Van Houdt et al. (2020), several versions of LSTM exist, but the most popular version is known as vanilla LSTM. A single LSTM unit is composed of a cell and three gates: input, output, and forget gate. The cell is responsible for keeping track of values over time; the input gate is responsible for combining the current input, the output of the previous LSTM unit, and the value of the cell in the previous iteration in order to decide whether to select the potential candidate values to be added. The forget gate considers current input, state on memory cells, and output at the previous time step in order to decide which information should be removed from previous cell states. The output gate computes the output (to be sent to the output block) by combining the output of that LSTM unit at the previous time, the current input, and the cell value in the previous iteration. LSTM networks are able to capture temporal information from time series data; since CNN networks are able to automatically extract significant features, combining both networks to build a hybrid architecture can bring benefits in terms of performance and accuracy in the HAR context.

3.2. Early exit networks

The concept of early exit (EE) networks was introduced by Teerapittayanon et al. (2017) where they envisaged a distributed deep neural network (DNN) over a hierarchically distributed computing infrastructure, i.e., end-devices, the edge/fog. and the cloud. Here, the idea was to consider small neural network (NN) models (running on devices or edge/fog) and larger NN models (running on the cloud), considering that also small NN models are capable of performing classification with a certain confidence. In such case, there is no need to rely on a further processing in the upper layers of the computing infrastructure, because a smaller portion of the DNN placed at the lower layers of the infrastructure is able to provide a satisfying classification with faster inference. The basic idea behind EE was actually first introduced in one of their prior work (Teerapittayanon et al., 2016), where they defined the "Exit" points as additional side branch classifiers. Here, a backbone DNN acting as the main branch is enhanced with multiple side branches, i.e., classifiers; such architecture leverages the experience that earlier layers are capable of learning features and performing inference of a large subset of the population correctly. Therefore, if a sample can be classified with high confidence and accuracy at the very early layers of the DNN, this allows for early exiting at these classifiers without the need of a deeper processing through all layers of the original DNN.

When dealing with EE architectures, it is necessary to consider several design aspects, such as: (i) how many exit points, and where to place them, (ii) how to structure the exit portion, (iv) how to train the network, and (v) the exit criteria, i.e., how to evaluate the confidence of the prediction. Our work hinges upon the discussion provided by recent literature (Scardapane et al., 2020), in which Scardapane et al. explore above mentioned aspects in detail. According to this discussion, where to place exit points is in general a combinatorial problem; however, in the case of one or two exit points, those are placed at 1/3 and 2/3 of the backbone network. The authors describe in detail the three main approaches that can be adopted, at training time, when a backbone network comprises early exits: joint training, layer-wise training, and independent training.

The joint training envisages the connection of a loss to each intermediate exit, i.e., classifier; then, those losses are properly combined and stochastic gradient descent is performed. Variations of such an approach exist, such for example merging predictions instead of losses. The layer-wise training is about layer optimization: at first, the goal is to minimize the loss of the first intermediate exit; then, as long as is needed, in an iterative fashion, a single intermediate exit is trained with the backbone layers preceding it. The independent training, also known as classifier-wise training, envisages the training of the backbone network as a first step, which is then completely frozen; subsequently, each of the intermediate exit networks is trained separately, as if those were stand-alone networks. Therefore, each sub-portion of the backbone network can be trained and considered independently of each other, and of the backbone itself, thus allowing to possibly select the early exit point. It is worth mentioning that, since the weight matrices of the backbone network are already learned and frozen, the loss function of the intermediate exit point depends only on the weight matrix of the intermediate classifier, thus reducing the uncoupled training (Baccarelli et al., 2020).

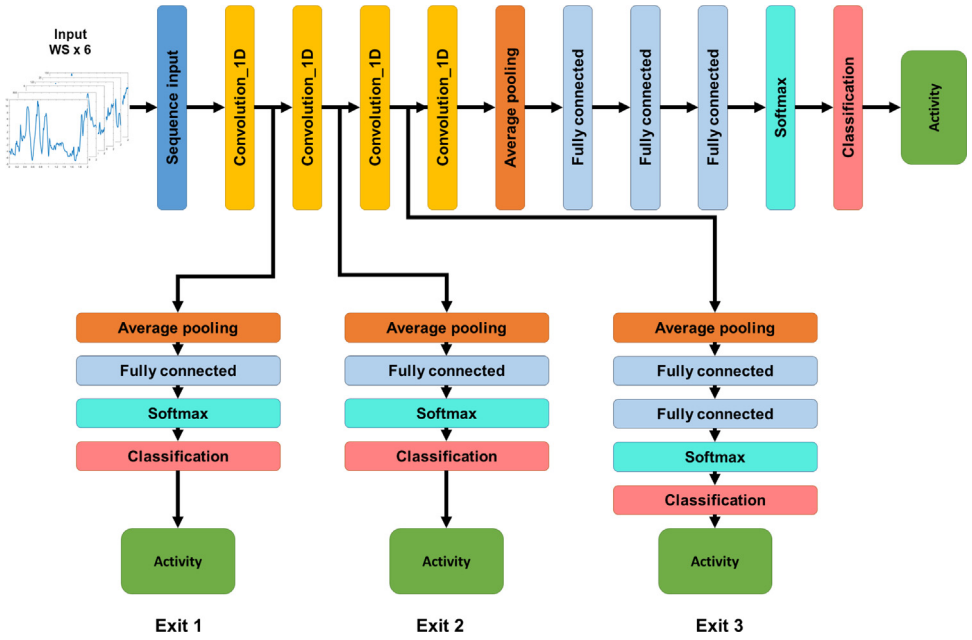


Fig. 1. The proposed 1D_CNN architecture.

Regarding the inference strategy in early exit networks, a suitable criteria metric in classification problems is network confidence. This criterion leverages the confidence of the network in its prediction to decide whether to exit or not at an early exit point. In Wang et al. (2017), three exit criteria are proposed: *(i)* probability-based, *(ii)* entropy-based, and *(iii)* risk minimization. The first two approaches have in common the capability to estimate the uncertainty of the model about the outputted prediction. However, the probability-based approach uses the probability over the predicted class to directly estimates uncertainty; whereas the entropy-based approach uses a well-calibrated class conditional probabilities to capture both overall model uncertainty and the certainty of the dominant class. The risk minimization approach envisages considering the computational cost of the intermediate exit point in the objective function. Such an approach allows therefore finding a trade-off between accuracy and computational cost. It is worth mentioning that both probability and entropy-based approaches aim at searching the so-called optima uncertainty threshold. The optimal uncertainty threshold guarantees the maximum accuracy such that further processing with a more complex model is not required.

4. Proposed methodology

In this section, we describe the methodology proposed to evaluate the effectiveness of the EE strategy in HAR. First of all, we present the network architectures and the used datasets. Then we illustrate the training and inferencing procedures.

4.1. Proposed multi-exit networks

The following neural network typologies have been chosen to be tested: (i) 1D_CNN, (ii) 2D_CNN, (iii) LSTM, and (iv) 1D_CNN–LSTM.

A key aspect in the design of multi-exit networks regards the placement of exit points. In fact, they can be positioned to be equally or variably spaced, where spacing is meant to be measured not (only) in terms of layer depth, rather in terms of the number floating point operations (FLOPs) performed by the network up to a given position (Laskaridis et al., 2021).

For each of the four type of networks analyzed and described below we placed three exit branches along the backbone of the network according to a logic of (almost) equally spaced placement.

Fig. 1 shows the architecture of the proposed 1D_CNN with its three exit points. The network takes in input the six signals gathered from the triaxial accelerometer and gyroscope appropriately divided into windows of size equal to WS. Clearly, deciding the size of the time window is a non-trivial task because the length of the time window impacts the performance of classification models (Banos et al., 2014). Indeed, it should be large enough to enable automatic recognition of the underlying human activity, but also not too large to include consecutive activities. For HAR tasks, different window lengths have been used in the literature, from 1 s up to 30 s (Lattanzi et al., 2022; Cheng et al., 2010; Hassan et al., 2018; Hou, 2020). In this study, we opted for a 3 s time window as a reasonable choice, given the characteristics of the adopted datasets (specified in Section 4.2), in terms of activities to be classified and of sampling ratios.

The segmented signals are arranged by a sequence input layer in a six-channel data chunk that is forwarded to the subsequent one-dimensional convolutional layers. In particular, the backbone network contains four consecutive convolutional layers followed by a single average pooling layer. Notice that, the output of each convolutional layer is processed by a ReLU and by a normalization layer that are not shown in the figure. The last part of the network is made up of three different layers of fully connected neurons that act as a multi-layer Perceptron. Also in this case, the three dropout layers which intersperse the neurons layer are not shown. The proposed network then terminates with a standard layer that computes the softmax function used by the classification layer to calculate the cross-entropy loss and to infer the activity label.

Starting from the backbone network just described, we proposed three exit points with increasing complexity. The *Exit 1*, just after the first convolutional layer, applies the average pooling layer followed by a single fully connected layer which produces the output for the softmax and the classification layer. The *Exit 2* includes the same layer but it takes as input the signals produced by the second convolutional layer. Finally, the *Exit 3* originates after the third convolutional layer and, unlike the first two points, it contains an extra fully connected layer.

Fig. 2 shows a completely similar network, with respect to the 1D_CNN, which makes use of the two-dimensional convolutional layers instead of the one-dimensional. The only difference is represented by the pre-processing phase which elaborates the input signals in order

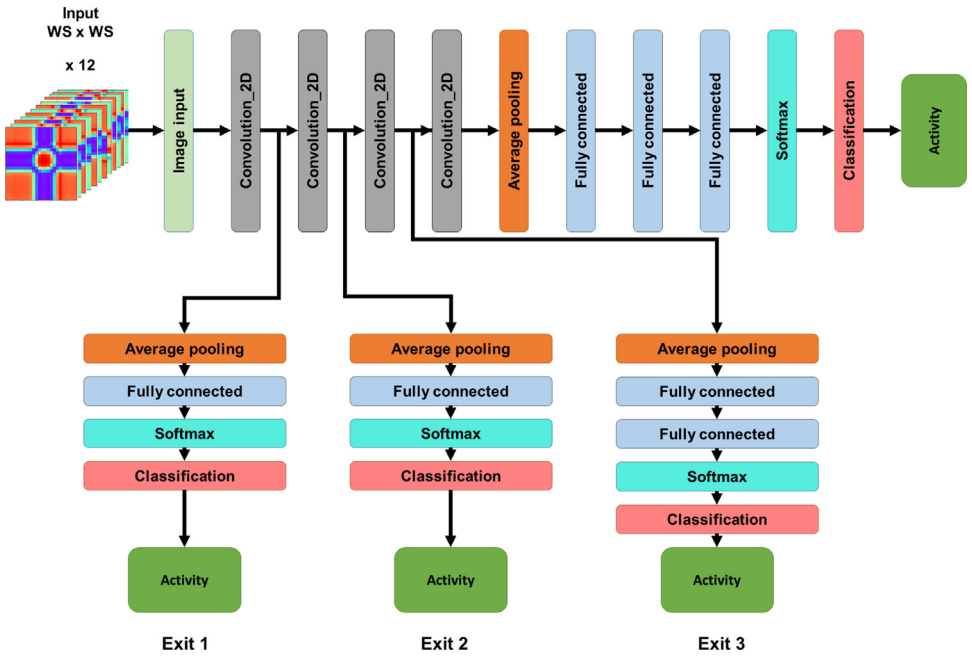


Fig. 2. The proposed 2D_CNN architecture.

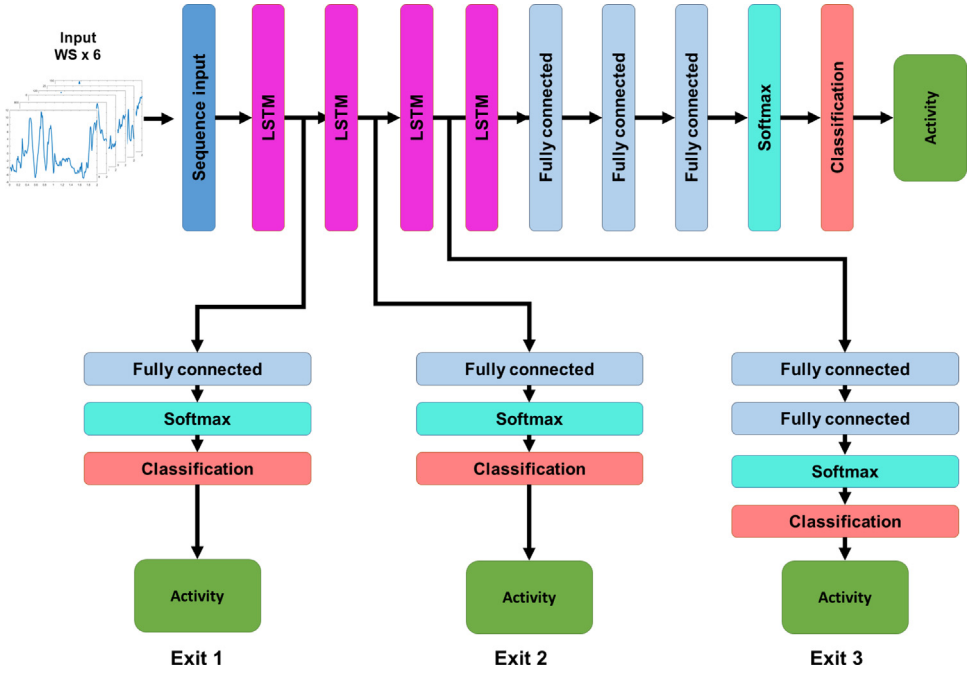


Fig. 3. The proposed LSTM architecture.

to obtain twelve square images as reported in Section 3. The produced images are then managed by an image input layer before going through the network structure. Notice that, also the three proposed exit points are completely identical with those used in the 1D_CNN.

In Fig. 3 a four LSTM layers network is represented. The LSTM stacked layers precede a three-layer Perceptron network followed by a softmax and by a standard classification layer. The three proposed exit points are, once again, identical to those we used in the previous networks. Notice that, both the LSTM layers and the Perceptron layers are interspersed with dropout layers not shown in the figure.

The architecture of the last proposed network is shown in Fig. 4. It represents a stacked convolutional-LSTM network that comprises three one-dimensional convolutional layers followed by two LSTM layers. The implemented exit points include a different number of convolutional, LSTM, and Perceptron layers so as to constitute points of increasing complexity.

In order to define the various hyperparameters that characterize the different models, we referred to some of the most current works in the literature. Starting from these, we carried out a manual tuning of the individual parameters trying to obtain the best performance in terms of classification accuracy. Table 1 reports, for each layer, the number of main elements that make it up. So for a convolutional layer, it represents the number of convolutional filters, for the LSTM layer the number of hidden units (i.e. the amount of information remembered between time steps), and for the fully connected layer the number of hidden neurons. Notice that, the last fully connected layer contains a

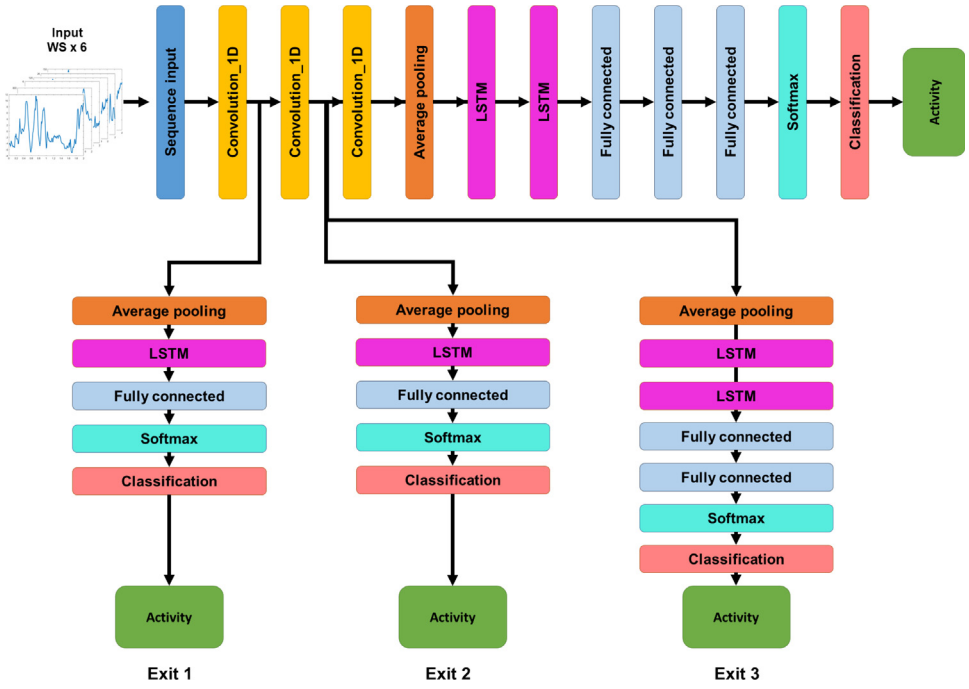


Fig. 4. The proposed 1D_CNN_LSTM architecture.

Table 1
Proposed models hyperparameters.

Model	Layer	#elements		Layer	#elements
1D_CNN	conv_1	128	2D_CNN	conv_1	128
	conv_2	256		conv_2	256
	conv_3	256		conv_3	256
	conv_4	256		conv_4	256
	fully conn_1	256		fully conn_1	256
	fully conn_2	128		fully conn_2	128
	fully conn_3	#classes		fully conn_3	#classes
LSTM	lstm_1	64	1D_CNN–LSTM	conv_1	128
	lstm_2	128		conv_2	256
	lstm_3	256		conv_3	256
	lstm_4	512		lstm_1	64
	fully conn_1	256		lstm_2	128
	fully conn_2	128		fully conn_1	256
	fully conn_3	#classes		fully conn_2	128
				fully conn_3	#classes

Table 2
Networks characterization.

	Size [MB]	Params [#]	Inf. latency [ms]	Accuracy
1D_CNN	10.36	2.76×10^6	60.6	0.970
LSTM	10.02	2.67×10^6	6.6	0.937
1D_CNN_LSTM	8.11	2.01×10^6	62.8	0.973
2D_CNN	36.16	33.48×10^6	77.5	0.939

number of hidden units that directly represent the number of classes of the classification problem so it is different for each used dataset.

4.2. Datasets

We evaluated the EE strategy on four different HAR datasets namely: (i) WISDM, (ii) REALWORLD16, (iii) UCI-HAR, (iv) OU-ISIR.

WISDM (Kwapisz et al., 2011): this dataset includes data collected from 51 subjects, each of whom was asked to perform 18 tasks for three minutes each. Data (sampled at 20 Hz) contains both accelerometer and gyroscope signals and were recorded using a smartwatch placed on the dominant hand and a smartphone placed on the pocket during the following activities: *walking, jogging, stairs, sitting, standing, typing, brushing teeth, eating soup, eating chips, eating pasta, drinking, eating a sandwich, kicking a soccer ball, playing tennis, dribbling, writing, clapping, and folding clothes*.

REALWORLD16 (Sztyler et al., 2017): the data are based on acceleration, GPS, gyroscope, light, magnetic field, and sound level collected at a sampling frequency of 50 Hz of the following activities: *climbing stairs down and up, jumping, lying, standing, sitting, running, and walking*. Each activity was performed for ten minutes (except for jumping which was repeated for about 2 min) by 15 subjects of different ages and gender in real-world conditions. Each subject was instrumented with different wearable sensors in order to capture signals generated in different body positions such as chest, forearm, head, shin, thigh, upper arm, and waist.

UCI-HAR (Reyes-Ortiz et al., 2016): it contains activity data gathered from 30 participants of varying ages, races, heights, and weights (aged between 18 and 48 years). The accelerometer and gyroscope signals are sampled at 50 Hz from a common smartphone placed at waist level. The monitored activities are: *walking, walking upstairs, walking downstairs, sitting, standing, and lying down*

OU-ISIR (Ngo et al., 2014): this is a very large dataset provided by the Institute of Scientific and Industrial Research (ISIR), of Osaka University (OU). It contains both accelerometer and gyroscope inertial measurements recorded at a sample rate of 50 Hz using a smartphone on 744 subjects (389 males and 355 females) with ages ranging from 2 to 78 years. This dataset is dedicated to the study of the different gait capabilities in age and gender of people and it contains only four different activities which are: *walking up on the slope, walking down on the slope, walking in flat (entering), and walking in flat (exiting)*.

4.3. Training the model

Training a EE network needs some considerations and design choices (see Section 3.2). This work implements the independent training of the backbone network and of each sub-networks defined by the three exit points. First of all, the raw datasets have been split into 75% for training and 25% for testing, then the training set has been used to train each network independently. Partitioning the entire dataset once and for all ensures that the backbone network and its sub-networks are

trained with the same samples so that there is no risk that during the inference phase, a sub-networks may have already encountered some of them. Once the models were trained, the classification performances of the backbone network and of the EE approach were assessed using the testing samples. To increase the robustness of the results, the entire procedure has been repeated five times, randomly choosing which samples to insert in the training set and which ones in the test set.

4.4. Making the inference

In the inference phase, the EE technique traditionally allows reducing the prediction latency thanks to the fact that the best exit point is dynamically chosen based on the classification "difficulty" of each sample. As reported in Section 3.2 several policies can be applied to drive the exit point selection. In this paper, we make use of the normalized entropy as a measure of the prediction confidence and we compare it with a user-selected threshold to decide whether to early exit the architecture. According to the work presented by Wang et al. in 2017 (Wang et al., 2017), denote by $c_{it}(x)$ the prediction of the ith exit point on the tth class, and by C the total number of classes. The normalized entropy of the prediction is:

$$H[c_i(x)] \triangleq \frac{1}{\log(C)} \sum_t c_{it}(x) \log(c_{it}(x)) \tag{1}$$

Defining a user-selected entropy threshold β_i for each exit point, we can apply the procedure described in Algorithm 1 to obtain the inference on a EE network. In particular, this procedure returns the *Effective Classification* ($EC(x)$) of a sample x for the chosen exit point and the *Effective Inference Latency* ($EIL(x)$) which is calculated as the sum of latency of the chosen exit point $l_i(x)$ with those of the exit points previously tested. Therefore, an EE network is more efficient the more often it can stop the inference phase early by testing a few exit points before finding the one that satisfies the confidence threshold. For instance, the best case occurs when the first exit point is already able to satisfy the confidence threshold. In this case, the $EIL(x)$ of the network will be equal to the latency of the first exit point $l_0(x)$. On the other hand, in the worst case, no available exit points satisfy the confidence threshold so the effective classification is done by the backbone network (the last exit point). In this condition, $EIL(x)$ is the sum of the inference latency of all available exit points on the network. The efficiency of an EE network can be evaluated by calculating the average EIL defined as:

$$\overline{EIL} = l_0 + \sum_{i=1}^{M} \alpha_i l_i \tag{2}$$

where α_i represents the exit ratio of the ith exit point. Obviously, α_i strictly depends on the selected β_i and on the prediction confidence of each exit point (i.e. the normalized entropy of the prediction).

5. Experimental setup

In this section, we provide a description of the performance metrics and of the experimental setup used to evaluate the effectiveness of the EE technique.

5.1. Performance metrics

Once the models have been trained on a specific dataset, we evaluate, both for the backbone network and for its EE version, the ability to correctly classify the human activities and the average network latency. Measuring the classification capability in a multi-class problem entails the evaluation of the following quantities for each of the N classes ($i \in [1 \cdots N]$ is an index that identifies a specific class): TP_i, the number of true positives predicted for class i; TN_i, the number of true negatives predicted for class i; FP_i, the number of false positives predicted for class i; FN_i, the number of false negatives predicted for class i.

Algorithm 1 Sample prediction on EE networks

M: number of exit points in the network
x : current sample
β_i : entropy threshold for the ith exit point
$c_i(x)$: classification of sample x by the ith exit point
$l_i(x)$: inference latency of the ith exit point
$EIL(x)$: effective inference latency of sample x
$EC(x)$: effective classification of sample x

$i \leftarrow 0$
$EIL(x) \leftarrow 0$
while $i < M$ **do**
  $EIL(x) \leftarrow EIL(x) + l_i(x)$
  $EC(x) \leftarrow c_i(x)$
  if $H[c_i(x)] \geq \beta_i$ **then**
    break
  end if
  $i \leftarrow i + 1$
end while
return $EIL(x)$, $EC(x)$

Subsequently, these indicators can be used to compute the following metrics (corresponding to the so called *macro-averaging* measures) (Sokolova and Lapalme, 2009):

$$Precision = \frac{1}{N} \sum_{i=1}^{N} \frac{TP_i}{TP_i + FP_i} \tag{3}$$

$$Recall = \frac{1}{N} \sum_{i=1}^{N} \frac{TP_i}{TP_i + FN_i} \tag{4}$$

$$F1score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{5}$$

$$Accuracy = \frac{1}{N} \sum_{i=1}^{N} \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i} \tag{6}$$

In order to calculate the performance of the EE version, first of all, we characterize each ith exit point of the model in terms of inference latency (l_i) ad normalized entropy of the prediction ($H[c_i(x)]$). Then, for a given entropy threshold (β_i), we applied the algorithm 1 described in Section 4.4 to compute the $EIL(x)$ and the $EC(x)$, for each sample x of the testing set, from which we can easily derive $\overline{EIL}$ and the metrics of the Eqs. (3) (4) (5) (6).

5.2. Setup configuration

The network models have been implemented and tested on Matlab2022a® platform running on a Windows® desktop pc equipped with an Intel® Core i9 and 16 GB of RAM.

The dataset records, composed of six distinct signals, have been divided into time windows and each of these has been considered as a sample to be used to train and test the classifiers. We refer the reader to Section 4.1 for the discussion about the size of the time window decision.

6. Experimental results

In this section, we describe the proposed experiments and report the corresponding results. First of all, we characterize the backbone networks in terms of classification performance and computational complexity. Then we compare these results with those obtained with the EE technique. Notice that, the reported results have been computed by repeating the training–testing procedure five times, randomly choosing which samples to insert in the training set and which ones in the test set.

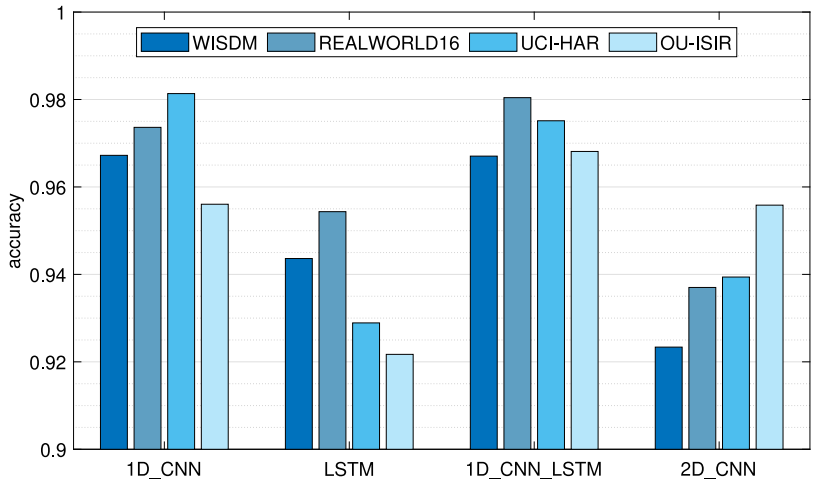


Fig. 5. Network classification accuracy.

6.1. Networks characterization

Fig. 5 shows the classification accuracy of the proposed backbone networks, without the application of the EE strategy, calculated over the four datasets. It is interesting to note that these networks obtain very good results in all datasets, achieving an accuracy that never drops below about 93%

Moreover, the 1D_CNN and the 1D_CNN_LSTM outperform other networks in all conditions, reaching a maximum accuracy of about 98% respectively in UCI-HAR and REALWORLD16 datasets. Notice that, precision, recall, and F1-score, although they have been calculated for each experiment, they are not reported on the figure since they do not show significant variations with respect to the accuracy which was therefore chosen to represent the classification performance of the models.

Table 2 reports a complete characterization of these networks in terms of size, number of trainable parameters, inference latency, and classification accuracy obtained by averaging the values measured in each dataset. For what concerns the model size, the 2D_CNN shows the largest memory occupation which is more than 3x with respect to the size of the others. This is also highlighted by the higher number of trainable parameters which reaches more than 33 million. From the inference latency point of view, the LSTM network is the fastest taking just 6 ms to classify a sample which is about an order of magnitude lower than the time spent by others. Finally, the average accuracy reported in the table confirms the supremacy of the 1D_CNN and the 1D_CNN_LSTM networks already seen in Fig. 5.

To better understand the relative performance of the four models, in Fig. 6, we represent each network in an inference latency Vs. classification loss Pareto plane. Here the 1D_CNN and the 1D_CNN_LSTM represent the best trade-off between classification loss and inference latency but it is equally evident that, in the face of a slight classification loss, the LSTM network entails a net advantage in terms of latency.

6.2. Early exit performance

To investigate the efficiency of the EE strategy applied to the proposed models we used the Algorithm 1 described in Section 4.4. Notice that, in order to reduce the degrees of freedom, in the following experiments we applied the same threshold at each exit point (i.e. we assumed that $\beta_i = B,\ \forall i \in [1, 2, \ldots, M]$). In these conditions, we iteratively executed the algorithm by varying B in the range [0,1] and we measured the corresponding $\overline{EIL}$.

Fig. 7 shows, for each network, the Pareto curves (classification loss Vs. $\overline{EIL}$) obtained when varying B on the different datasets with highlighted, by means of red circles, some representative values of the entropy threshold B. When B is equal to zero, the network always stops the inference at the first exit but, on the other hand, setting it to 1 does not entail always exiting at the backbone network. In fact, even with the threshold set at the maximum value, the network can still stop at early exits if the prediction confidence for that sample reaches the maximum value. Obviously, the greater the B, the greater the likelihood of using the furthest exits, and potentially, the effectiveness of the network. On the other hand, using more and more far exits, while increasing the effectiveness in terms of accuracy, decreases the efficiency in terms of prediction latency.

Interestingly, all the networks show a similar trend that identifies as the best trade-off some points given by a value of B in the range of about 0.80 to 0.90 and that, using an even higher threshold, involves a high decrease of efficiency in the face of a modest increase in effectiveness. Another interesting result is that even using the maximum value of B, the $\overline{EIL}$ never reaches the inference latency of the respective full network while maintaining about the same effectiveness.

The comparison between the EE strategy and the baseline (the corresponding backbone) is reported in Tables 3 and 4 which compare, respectively, the effectiveness and the efficiency of the networks. In particular, Table 3 shows the accuracy variation between the baseline and the EE approach when using the maximum (max) and the optimal (opt) value of B (i.e., respectively, 1 and 0.85). In the case of using max value, the performances are practically identical to those of the baseline or even better (negative values). Only for the 2D_CNN a worse in accuracy can be noticed of about 3% when using the OU-ISIR dataset. The results change slightly when the optimal threshold (opt) value is used. In this case, we recorded a maximum loss of about 8% again for the 2D_CNN on OU-ISIR, while, in the case of 1D_CNN and 1D_CNN_LSTM, the obtained results are practically identical to the baseline with loss of accuracy never greater than 0.4%. Finally, also the LSTM network, except for OU-ISIR, where it loses about 3%, obtains results in line with those of the baseline.

Table 4 shows the inference latency gain of the EE approach with respect to the baseline. Both for max and opt configurations, the EE approach registered a positive gain, demonstrating a significant increase in efficiency. For instance, using the max threshold, no model falls below an average gain of about 2× with a maximum of about 6× for the 1D_CNN_LSTM. Reducing the threshold to the optimal value (i.e. to about 0.85) the latency gain increases for all networks reaching the maximum average value of more than 20× (with a peak of 35×) using the 1D_CNN_LSTM network.

Analyzing the results of the two tables together, it appears that there are no conditions in which the application of the EE technique is disadvantageous in terms of accuracy, despite significantly increasing efficiency, if the max value of the threshold is used. On the other hand,

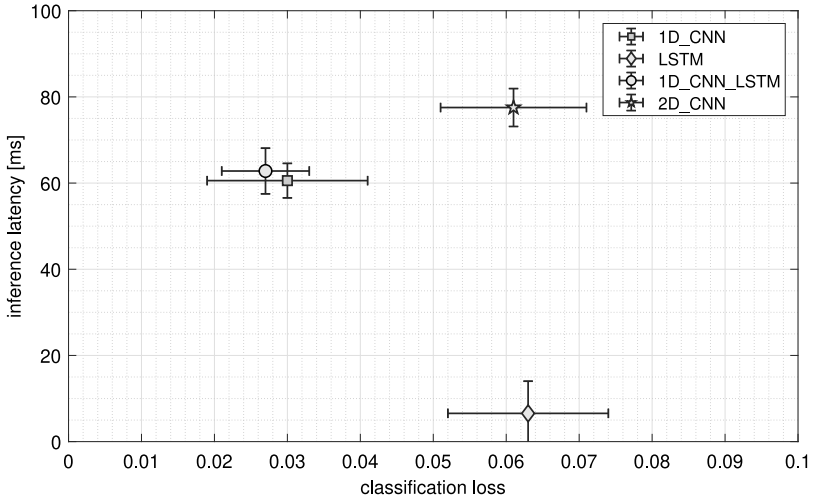


Fig. 6. Network latency Vs. classification loss.

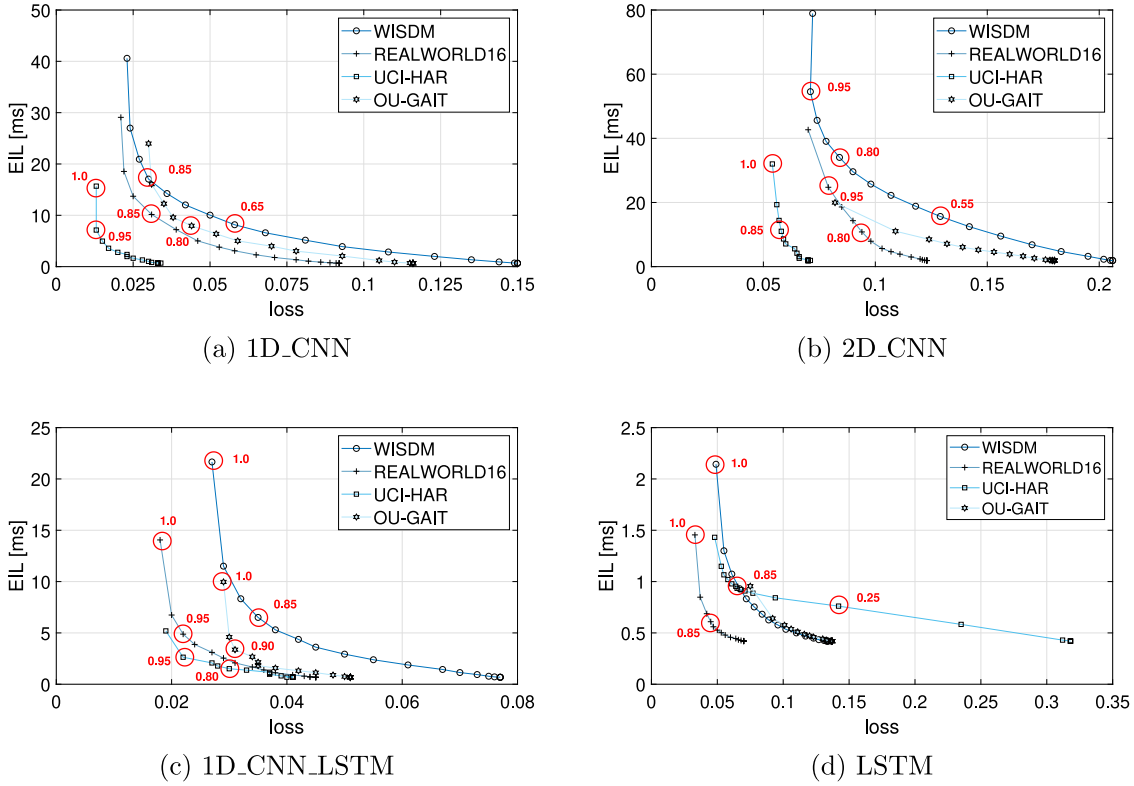


Fig. 7. Pareto curve reporting the accuracy loss Vs. the expected inference latency ($\overline{EIL}$). The red circles highlight some representative values of the entropy threshold B.

Table 3
Classification accuracy variation of the EE approach with respect to the baseline for maximum (*max*) and optimal (*opt*) cross-entropy threshold.

Dataset	1D_CNN		2D_CNN		LSTM		1D_CNN_LSTM	
	max	opt	max	opt	max	opt	max	opt
WISDM	−0.009	−0.004	−0.005	0.002	−0.007	0.010	−0.005	0.002
REALWORLD16	−0.004	0.003	0.007	0.027	−0.013	−0.001	−0.002	0.004
UCI-HAR	−0.005	−0.001	−0.006	−0.003	−0.023	−0.013	−0.006	0.003
OU-ISIR	−0.012	−0.005	0.038	0.088	−0.003	0.027	−0.003	0.002
Average	−0.008	−0.002	0.008	0.028	−0.012	0.006	−0.004	0.003

by reducing the threshold to *opt*, a further improvement in efficiency is obtained without substantially reducing the classification capability except in the case of the 2D_CNN network.

We chose the 1D_CNN network and the REALWORLD16 dataset as a representative benchmark, to deeply analyze the performance of the EE technique and to highlight its peculiarity. Fig. 8 shows the confusion

Table 4
Inference latency gain of the EE approach with respect to the baseline for maximum (*max*) and optimal (*opt*) cross-entropy threshold.

Dataset	1D_CNN		2D_CNN		LSTM		1D_CNN_LSTM	
	max	opt	max	opt	max	opt	max	opt
WISDM	2.297	5.983	0.982	1.984	3.058	7.047	2.900	9.650
REALWORLD16	3.019	9.300	1.816	5.408	4.503	10.757	4.467	16.196
UCI-HAR	7.118	26.420	2.421	7.035	4.580	6.427	12.088	35.338
OU-ISIR	3.180	8.238	3.884	10.882	6.865	12.179	6.300	23.474
Average	3.903	12.485	2.276	6.327	4.751	9.103	6.439	21.165

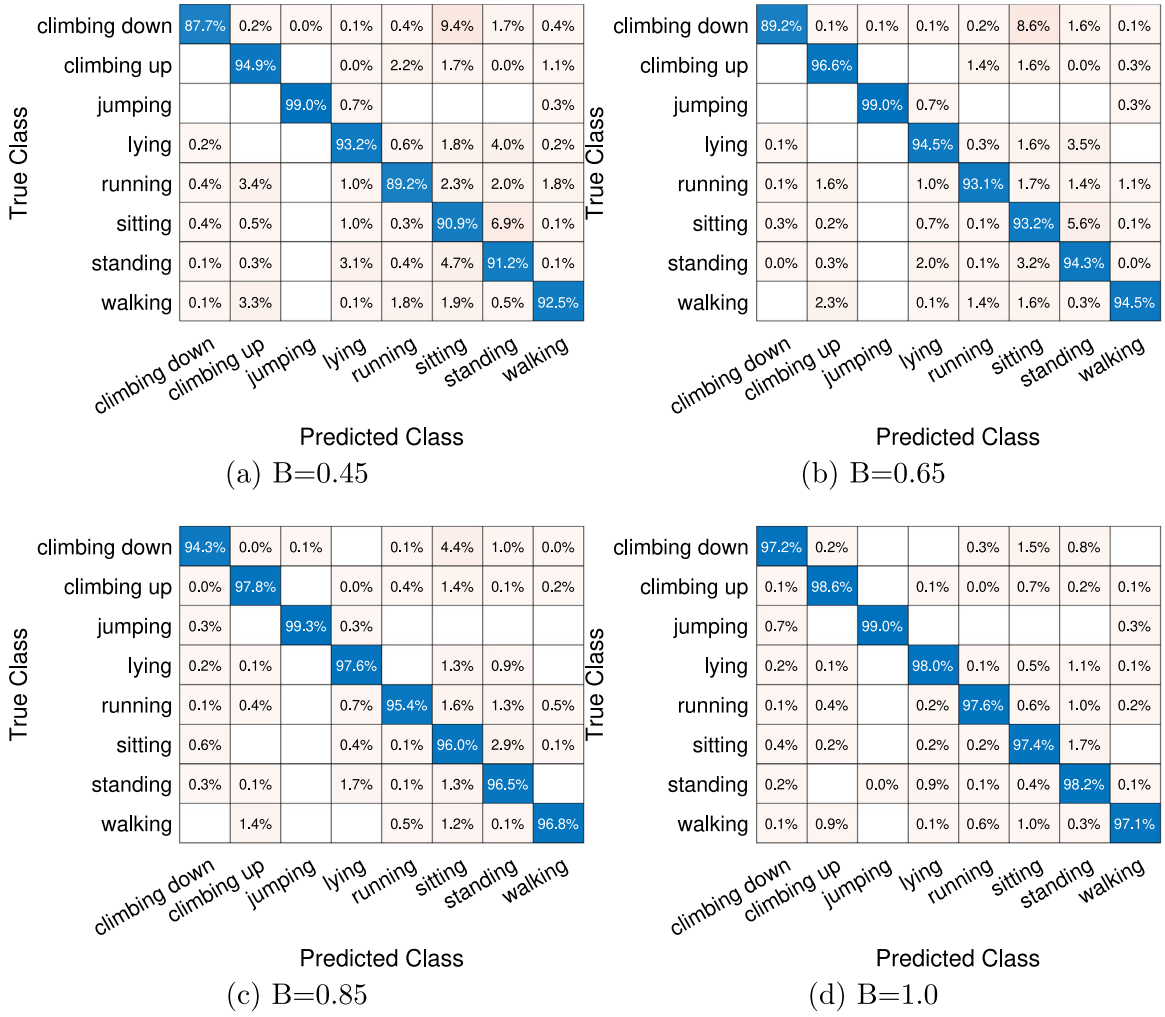


Fig. 8. The expected classification matrices obtained with four different values of the threshold B for the LSTM network on the REALWORLD16 dataset.

matrices, calculated on top of the $EC(x)$ results, obtained using four different values of B namely $0.45, 0.65, 0.85$, and 1.0. The matrices clearly show that some activities are well classified also with low values of the threshold ($B = 0.45$) leading to a high probability of exit at early stages, such as, for instance, "climbing up", "jumping", and "lying". On the other hand, some other needs higher thresholds which involve the furthest exit points. The "climbing down" activity, for instance, is well classified using the maximum threshold value ($B = 1.0$) while in the other cases, it is resolved as "sitting" with a non negligible frequency.

In order to further investigate the behavior of the system, we focused on the value of $B = 0.85$ and, for this, we counted the number of times each activity determined the exit at a particular point. The exit ratio of the activities are reported in Fig. 9. Interestingly, different activities show strongly different behaviors: the easiest to classify are discovered in the early stages while the hardest require the involvement of the furthest exits. For instance, the "jumping" activity exits at about 98% at the first point while "sitting" and "standing" no more than 47% and they arrive to involve the backbone for more than 25% of the evaluated samples. Another interesting finding is represented by the fact that, for instance, despite an almost equal exit rate at the first stage of activities "climbing down" and "lying", they involve the furthest exit points in quite different ways. In fact, the former uses the backbone exit more frequently while the latter is adequately classified already in the second exit point reducing the use of the more complex classifier.

This in-depth analysis clearly highlights the fundamentals of the EE technique which bases its advantage on the fact that some samples are

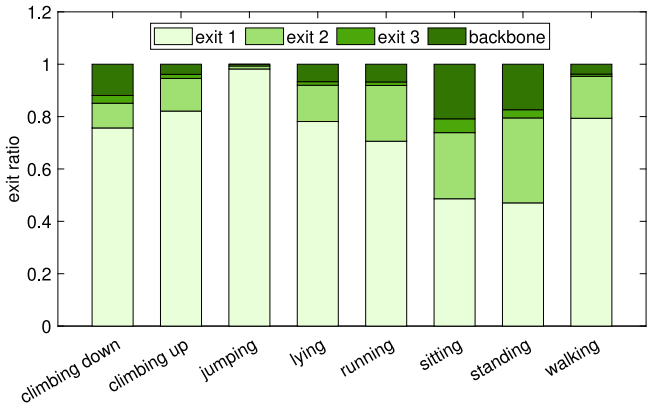


Fig. 9. Exit ratio at each point for the activities in the REALWORLD16 dataset for the 1D_CNN network obtained with $B = 0.85$.

easier to classify and therefore can leave the pipeline sooner, reducing the use of more complex and expensive models.

6.3. Non-parametric significance test

To statistically assess whether the accuracies of the EE networks are different from the corresponding backbone, a statistical McNemar test with a confidence level of 95% has been performed (Fagerland et al., 2013). In particular, we have tested the null hypothesis that the two classifiers have equal accuracy for predicting the true classes.

Fig. 10 reports the p-value computed by the statistical test when varying the threshold B of the EE network for each dataset. Notice that, for p-values greater than 0.05 the null hypothesis cannot be rejected under a confidence level of 95%. As expected, for lower values of B the EE networks cannot be considered equivalent with respect to the corresponding backbone. Moreover, further increasing B corresponds to an increase in the p-value that, for B around 0.5–0.7 (it depends on the network), involves having to consider the two models as identical from a statistical point of view (i.e. p-value ≥ 0.05). On the other hand, it seems interesting that beyond a certain value of B the p-value falls again and, in almost all cases, in the last points of the graph the null hypothesis must be rejected.

While the non-equivalence for very low values of B is evident that it is due to the reduced classification capacity of the EE network, the rejection of the null hypothesis for high values, probably, must be due to the fact that the EE network performs better than the backbone as reported in previous results.

6.4. Adding the communication latency

The experiments presented in the previous section are based on the assumption that the backbone network, with its exit points, is installed in the memory of the same device and that each branch to an exit point can be executed locally without introducing any latency due to the data transfer. In a distributed environment, where, for instance, each network branch is stored and executed in a different device, a communication latency should be paid each time that the current exit point does not satisfy the entropy threshold.

The magnitude of the communication latency may or may not make the EE technique ineffective. To this purpose, we call *breakpoint latency* the value of the communication latency to be paid at each exit point, following the first, for which the $\overline{EIL}$ of an EE network exceeds the original average backbone latency (baseline) for any possible entropy threshold. This means that, for a communication channel with a latency equal to or greater than the breakpoint latency, there are no threshold values that can make the EE strategy effective.

Table 5
Case study parameters.

	Transfer rate [Mbps]	Latency [ms]	Sending time [ms]
PAN (Bluetooth)	1	100	114.06
LAN (WiFi)	54	10	10.26
Internet (Mix)	20	20	20.70

Fig. 11 shows the estimated *breakpoint latency* for the proposed models tested on the four datasets. Notice that, for sake of simplicity, the latency was assumed to be the same in each communication channel between the devices hosting the different exit points. The 1D_CNN and the 1D_CNN_LSTM result in a significantly higher *breakpoint latency* with respect to the LSTM and 2D_CNN networks. This implies that the EE technique, applied to the first two networks, shows greater robustness such as making it convenient to use even when the communication should be a bottleneck. For instance, an EE 1D_CNN_LSTM is still convenient for communication latency values that can reach up to about 400 ms. On the other hand, a latency as low as 20 ms or 30 ms can make it useless to apply EE in LSTM or 2D_CNN networks.

From these results, it is evident that a distributed EE technique may not be convenient not only for those models that are already very fast, in which the inference latency can be as equal as, or even lower than, that of communication (LSTM), but also in those which although showing long inference times (2D_CNN), are forced to make extensive use of far exit points to satisfy the accuracy requirements.

In order to evaluate the suitability of the EE technique in more realistic conditions, we simulate a case study proper to HAR applications by assuming a setup in which the deep model is spread through four devices connected by means of different network technologies. In particular, a wearable device, which acts as a sensor by collecting accelerometer and gyroscope signals, holds the first exit point and it is connected, through a Bluetooth-enabled PAN, to a gateway that contains the second exit point. The gateway can easily be represented by a smartphone connected to a WiFi LAN. In turn, in the LAN there may be a router capable of hosting the third exit point and which can reach the cloud, by means of the Internet network, where it can find the backbone network (the last exit point).

Table 5 reports the typical transfer rate, latency, and the resulting sending time of our case study obtained by analyzing the current scientific literature (Baliga et al., 2011; Dusza et al., 2013). The sending time has been calculated starting from the latency and transfer rate assuming a payload composed of 6 signals sampled at 50 Hz with 16 bits for sample resulting in about 4.8 KB for a processing window lasting one second.

To highlight the suitability of the application of the EE technique in the proposed case study, we compared, in the Pareto loss Vs. accuracy plan, the backbone network with the EE strategy (we took the REALWORLD16 dataset as a case-study).

In particular, Fig. 12 shows the reciprocal positioning of the backbone networks (orange points) with respect to the EE curves obtained for all possible values of B. Notice that, a curve positioned above the backbone point identifies a deep model in which it is not advisable to use the EE technique (2D_CNN and LSTM in the figure) while, in the other cases (1D_CNN and 1D_CNN_LSTM), it still carries an advantage.

Considering the assumptions made, this case study shows that not all deep networks can benefit from the EE technique in the HAR application domain and that its suitability must be evaluated case by case regarding the performance of the model and the characteristics of the setup.

7. Conclusion

This paper proposes an empirical investigation of early exit deep neural networks in human activity recognition tasks across four different neural networks. The extensive experiments conducted on four

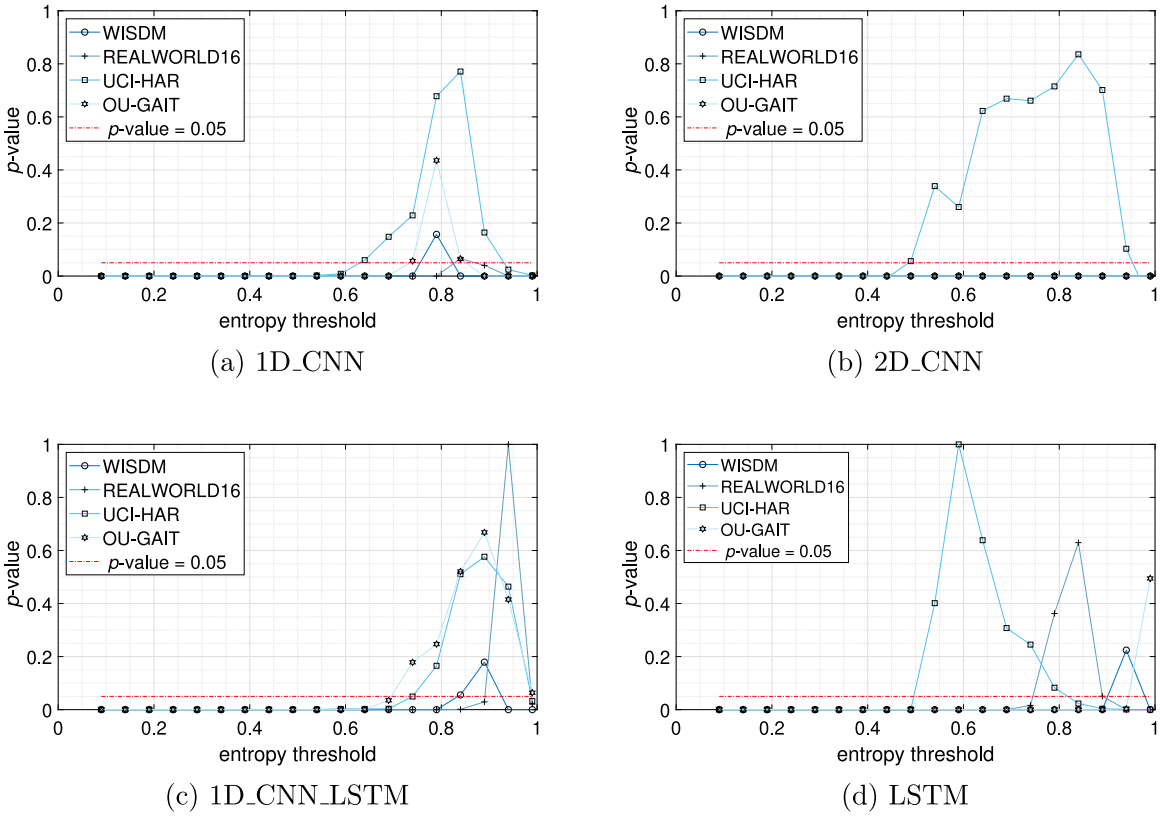


Fig. 10. p-value of the McNemar test when varying the entropy threshold B.

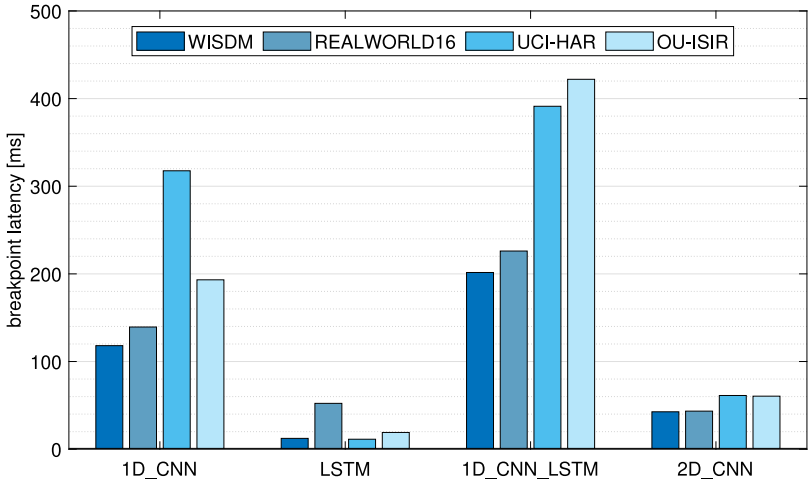


Fig. 11. Breakpoint communication latency.

publicly available datasets, confirm the high performance achieved by the backbone networks in the classification tests and outline the supremacy of 1D_CNN and 1D_CNN_LSTNM with respect to 2D_CNN and LSTM networks.

Adding the EE strategy locally shows that there are no conditions in which it is disadvantageous in terms of accuracy, except in the case of the 2D CNN network, despite an increased efficiency up to 35×. On the other hand, in a distributed environment, for instance where each network branch is stored and executed in a different device, and where a communication latency should be paid when you skip to the next exit point, our results point out that applying the EE technique may not always be convenient. In particular, for both the case of models that are already very fast such as LSTM, and also for those that, although showing long inference times (2D CNN), are forced to make extensive use of far exit points to satisfy the accuracy requirements, the adoption of EE can be disadvantageous.

In future steps, we plan to expand our analysis by considering different performance metrics in addition to latency and accuracy such as, for example, energy consumption. For this purpose, we plan to employ resource-constrained devices, such as ESP32 MCU, representing

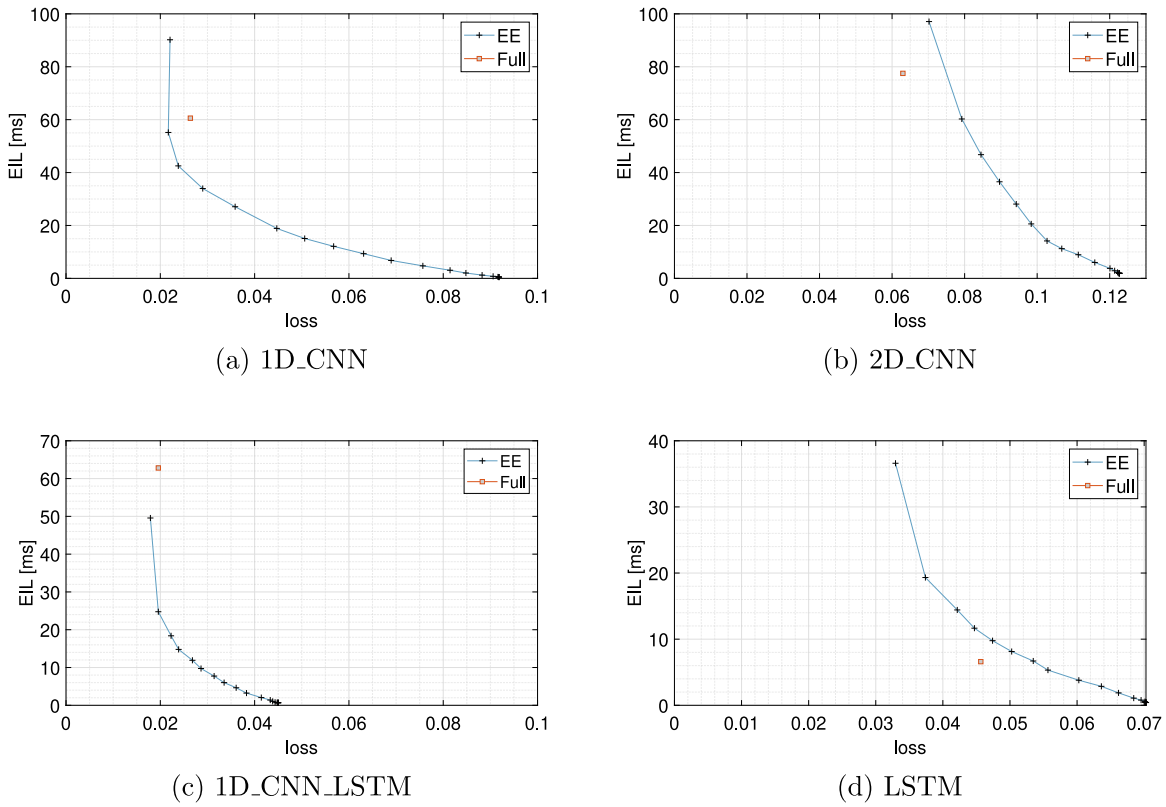


Fig. 12. Pareto curve loss Vs. latency for the case study.

an edge device, and a Raspberry Pi, in the role of a gateway, to in-depth characterize the EE suitability in a typical Internet of Things HAR application.

CRediT authorship contribution statement

Emanuele Lattanzi: Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Writing – original draft, Writing – review & editing. **Chiara Contoli:** Conceptualization, Writing – original draft, Writing – review & editing. **Valerio Freschi:** Conceptualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We used public datasets.

Acknowledgment

Chiara Contoli is a researcher co-funded by the European Union - PON Research and Innovation 2014-2020.

References

Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M., Farhan, L., 2021. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. J. Big Data 8 (1), 1–74.

Baccarelli, E., Scardapane, S., Scarpiniti, M., Momenzadeh, A., Uncini, A., 2020. Optimized training and scalable implementation of conditional deep neural networks with early exits for fog-supported IoT applications. Inform. Sci. 521, 107–143.

Baldini, G., Steri, G., Giuliani, R., Gentile, C., 2017. Imaging time series for internet of things radio frequency fingerprinting. In: 2017 International Carnahan Conference on Security Technology (ICCST). IEEE, pp. 1–6.

Baliga, J., Ayre, R., Hinton, K., Tucker, R.S., 2011. Energy consumption in wired and wireless access networks. IEEE Commun. Mag. 49 (6), 70–77.

Banbury, C., Zhou, C., Fedorov, I., Matas, R., Thakker, U., Gope, D., Janapa Reddi, V., Mattina, M., Whatmough, P., 2021. Micronets: Neural network architectures for deploying tinyml applications on commodity microcontrollers. Proc. Mach. Learn. Syst. 3, 517–532.

Banos, O., Galvez, J.-M., Damas, M., Pomares, H., Rojas, I., 2014. Window size impact in human activity recognition. Sensors 14 (4), 6474–6499.

Bhat, G., Tran, N., Shill, H., Ogras, U.Y., 2020. w-HAR: An activity recognition dataset and framework using low-power wearable devices. Sensors 20 (18), 5356.

Bonato, V., Bouganis, C.-S., 2021. Class-specific early exit design methodology for convolutional neural networks. Appl. Soft Comput. 107, 107316.

Cheng, J., Amft, O., Lukowicz, P., 2010. Active capacitive sensing: Exploring a new wearable sensing modality for activity recognition. In: International Conference on Pervasive Computing. Springer, pp. 319–336.

Cheng, X., Zhang, L., Tang, Y., Liu, Y., Wu, H., He, J., 2022. Real-time human activity recognition using conditionally parametrized convolutions on mobile and wearable devices. IEEE Sens. J. 22 (6), 5889–5901.

Deep, S., Zheng, X., 2019. Hybrid model featuring CNN and LSTM architecture for human activity recognition on smartphone sensor data. In: 2019 20th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT). IEEE, pp. 259–264.

Dusza, B., Ide, C., Cheng, L., Wietfeld, C., 2013. Copomo: a context-aware power consumption model for LTE user equipment. Trans. Emerg. Telecommun. Technol. 24 (6), 615–632.

Fagerland, M.W., Lydersen, S., Laake, P., 2013. The McNemar test for binary matched-pairs data: mid-p and asymptotic are better than exact conditional. BMC Med. Res. Methodol. 13 (1), 1–8.

Fedorov, I., Adams, R.P., Mattina, M., Whatmough, P., 2019. Sparse: Sparse architecture search for CNNs on resource-constrained microcontrollers. Adv. Neural Inf. Process. Syst. 32.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep learning.

Gu, F., Chung, M.-H., Chignell, M., Valaee, S., Zhou, B., Liu, X., 2021. A survey on deep learning for human activity recognition. ACM Comput. Surv. 54 (8), 1–34.

Hammerla, N.Y., Halloran, S., Plötz, T., 2016. Deep, convolutional, and recurrent models for human activity recognition using wearables. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence. AAAI Press, pp. 1533–1540.

Hassan, M.M., Uddin, M.Z., Mohamed, A., Almogren, A., 2018. A robust human activity recognition system using smartphone sensors and deep learning. Future Gener. Comput. Syst. 81, 307–313.

Hou, C., 2020. A study on IMU-based human activity recognition using deep learning and traditional machine learning. In: 2020 5th International Conference on Computer and Communication Systems (ICCCS). IEEE, pp. 225–234.

Huang, W., Zhang, L., Wu, H., Min, F., Song, A., 2022. Channel-equalization-HAR: A light-weight convolutional neural network for wearable sensor based human activity recognition. IEEE Trans. Mob. Comput..

Kwapisz, J.R., Weiss, G.M., Moore, S.A., 2011. Activity recognition using cell phone accelerometers. ACM SIGKDD Explor. Newslett. 12 (2), 74–82.

Laskaridis, S., Kouris, A., Lane, N.D., 2021. Adaptive inference through early-exit networks: Design, challenges and directions. In: Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning. pp. 1–6.

Laskaridis, S., Venieris, S.I., Almeida, M., Leontiadis, I., Lane, N.D., 2020. SPINN: synergistic progressive inference of neural networks over device and cloud. In: Proceedings of the 26th Annual International Conference on Mobile Computing and Networking. pp. 1–15.

Lattanzi, E., Donati, M., Freschi, V., 2022. Exploring artificial neural networks efficiency in tiny wearable devices for human activity recognition. Sensors 22 (7), 2637.

Lattanzi, E., Freschi, V., 2020. Evaluation of human standing balance using wearable inertial sensors: a machine learning approach. Eng. Appl. Artif. Intell. 94, 103812.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444.

Li, E., Zeng, L., Zhou, Z., Chen, X., 2019. Edge AI: On-demand accelerating deep neural network inference via edge computing. IEEE Trans. Wireless Commun. 19 (1), 447–457.

Ma, H., Li, W., Zhang, X., Gao, S., Lu, S., 2019. AttnSense: Multi-level attention mechanism for multimodal human activity recognition. In: IJCAI. pp. 3109–3115.

Matsubara, Y., Levorato, M., Restuccia, F., 2021. Split computing and early exiting for deep learning applications: Survey and research challenges. ACM Comput. Surv..

Mekruksavanich, S., Jitpattanakul, A., 2021. LSTM networks using smartphone data for sensor-based human activity recognition in smart homes. Sensors 21 (5), 1636.

Ngo, T.T., Makihara, Y., Nagahara, H., Mukaigawa, Y., Yagi, Y., 2014. The largest inertial sensor-based gait database and performance evaluation of gait-based personal authentication. Pattern Recognit. 47 (1), 228–237.

Odema, M., Rashid, N., Al Faruque, M.A., 2021. Eexnas: Early-exit neural architecture search solutions for low-power wearable devices. In: 2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED). IEEE, pp. 1–6.

Ordóñez, F.J., Roggen, D., 2016. Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. Sensors 16 (1), 115.

Pacheco, R.G., Bochie, K., Gilbert, M.S., Couto, R.S., Campista, M.E.M., 2021. Towards edge computing using early-exit convolutional neural networks. Information 12 (10), 431.

Perez-Pozuelo, I., Spathis, D., Clifton, E.A., Mascolo, C., 2021. Wearables, smartphones, and artificial intelligence for digital phenotyping and health. Digit. Health 33–54.

Qin, Z., Zhang, Y., Meng, S., Qin, Z., Choo, K.-K.R., 2020. Imaging and fusing time series for wearable sensor-based human activity recognition. Inf. Fusion 53, 80–87.

Rashid, N., Demirel, B.U., Al Faruque, M.A., 2022. AHAR: Adaptive CNN for energy-efficient human activity recognition in low-power edge devices. IEEE Internet Things J..

Reiss, A., Stricker, D., 2012. Introducing a new benchmarked dataset for activity monitoring. In: 2012 16th International Symposium on Wearable Computers. IEEE, pp. 108–109.

Reyes-Ortiz, J.-L., Oneto, L., Samà, A., Parra, X., Anguita, D., 2016. Transition-aware human activity recognition using smartphones. Neurocomputing 171, 754–767.

Roggen, D., Calatroni, A., Rossi, M., Holleczek, T., Förster, K., Tröster, G., Lukowicz, P., Bannach, D., Pirkl, G., Ferscha, A., et al., 2010. Collecting complex activity datasets in highly rich networked sensor environments. In: 2010 Seventh International Conference on Networked Sensing Systems. INSS, IEEE, pp. 233–240.

Samie, F., Bauer, L., Henkel, J., 2020. Hierarchical classification for constrained IoT devices: A case study on human activity recognition. IEEE Internet Things J. 7 (9), 8287–8295.

Scardapane, S., Scarpiniti, M., Baccarelli, E., Uncini, A., 2020. Why should we add early exits to neural networks? Cogn. Comput. 12 (5), 954–966.

Sokolova, M., Lapalme, G., 2009. A systematic analysis of performance measures for classification tasks. Inf. Process. Manage. 45 (4), 427–437.

Stisen, A., Blunck, H., Bhattacharya, S., Prentow, T.S., Kjærgaard, M.B., Dey, A., Sonne, T., Jensen, M.M., 2015. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems. pp. 127–140.

Sztyler, T., Stuckenschmidt, H., Petrich, W., 2017. Position-aware activity recognition with wearable devices. Pervasive Mob. Comput. 38, 281–295.

Tan, X., Li, H., Wang, L., Huang, X., Xu, Z., 2021. Empowering adaptive early-exit inference with latency awareness. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35. pp. 9825–9833.

Tang, Y., Zhang, L., Min, F., He, J., 2022. Multi-scale deep feature learning for human activity recognition using wearable sensors. IEEE Trans. Ind. Electron..

Teerapittayanon, S., McDanel, B., Kung, H.-T., 2016. Branchynet: Fast inference via early exiting from deep neural networks. In: 2016 23rd International Conference on Pattern Recognition (ICPR). IEEE, pp. 2464–2469.

Teerapittayanon, S., McDanel, B., Kung, H.-T., 2017. Distributed deep neural networks over the cloud, the edge and end devices. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). IEEE, pp. 328–339.

Van Houdt, G., Mosquera, C., Nápoles, G., 2020. A review on the long short-term memory model. Artif. Intell. Rev. 53 (8), 5929–5955.

Wang, J., Chen, Y., Hao, S., Peng, X., Hu, L., 2019. Deep learning for sensor-based activity recognition: A survey. Pattern Recognit. Lett. 119, 3–11.

Wang, X., Luo, Y., Crankshaw, D., Tumanov, A., Yu, F., Gonzalez, J.E., 2017. Idk cascades: Fast deep learning by learning not to overthink. arXiv preprint arXiv:1706.00885.

Wang, Z., Oates, T., 2015. Imaging time-series to improve classification and imputation. In: Twenty-Fourth International Joint Conference on Artificial Intelligence.

Xia, K., Huang, J., Wang, H., 2020. LSTM-CNN architecture for human activity recognition. IEEE Access 8, 56855–56866.

Xu, C., Chai, D., He, J., Zhang, X., Duan, S., 2019. InnoHAR: A deep neural network for complex human activity recognition. IEEE Access 7, 9893–9902.

Zappi, P., Lombriser, C., Stiefmeier, T., Farella, E., Roggen, D., Benini, L., Tröster, G., 2008. Activity recognition from on-body sensors: accuracy-power trade-off by dynamic sensor selection. In: European Conference on Wireless Sensor Networks. Springer, pp. 17–33.

Zhao, Z., Barijough, K.M., Gerstlauer, A., 2018. Deepthings: Distributed adaptive deep learning inference on resource-constrained IoT edge clusters. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 37 (11), 2348–2359.