# Energy efficiency of Deep Learning Compression Techniques in Wearable Human Activity Recognition

Chiara Contoli[1][0000−0003−2389−2593] and Emanuele Lattanzi[1][0000−0002−6568−8470]

Department of Pure and Applied Sciences, University of Urbino, Piazza della Repubblica 13, 61029 Urbino, IT
{chiara.contoli,emanuele.lattanzi}@uniurb.it

**Abstract.** Deploying deep learning (DL) models onto low-power devices for Human Activity Recognition (HAR) purposes is gaining momentum because of the pervasive adoption of wearable sensor devices. However, the outcome of such deployment needs exploration not only because the topic is still in its infancy, but also because of the wide combination between low-power devices, deep models, and available deployment strategies. We have investigated the outcome of the application of three compression techniques, namely lite conversion, dynamic quantization, and full-integer quantization, that allow the deployment of deep models on low-power devices. This paper describes how those three compression techniques impact accuracy and energy consumption on an ESP32 device. In terms of accuracy, the full-integer technique incurs an accuracy drop between 2% and 3%, whereas the dynamic quantization and the lite conversion result in a negligible accuracy drop. In terms of power efficiency, dynamic and full-integer quantization allow for saving almost 30% of energy. The adoption of one of those two quantization techniques is recommended to obtain an executable network model, and we advise the adoption of the dynamic quantization given the negligible accuracy drop.

**Keywords:** Human Activity Recognition · Compression techniques · Deep learning · Microcontroller
[1]

## 1 Introduction

Deep Learning (DL) has become a ubiquitous tool in many modern applications such as computer vision, natural language processing, and speech recognition. However, the computational requirements of DL models are often substantial, hindering their deployment on resource-constrained devices such as mobile

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

phones or embedded systems. As such, there has been significant research interest in developing efficient techniques to compress DL models without sacrificing their performance.

One approach to address this challenge is to leverage model compression techniques such as pruning[1], quantization[11], and knowledge distillation[9]. These techniques aim to reduce the number of parameters or operations in a DL model while maintaining its accuracy.

Pruning and quantization have been recently surveyed in terms of methods of compression and mathematical formulation by Liang et al. [12]. Pruning consists of removing unnecessary parameters or neurons, and connections because do not provide a significant contribution to resulting accuracy. As of today, pruning can be distinguished depending on various aspects that are considered during the operation. In particular, three categories exist: i) whether the pruned network is symmetric or not, it is classified as structured or unstructured pruning; ii) based on the pruned element type, it is classified as neurons or connections pruning; iii) based on when pruning steps are carried out, i.e., after training but before inference, or during the inference process, it is classified as static pruning or dynamic pruning, respectively.

Quantization, on the other hand, consists of reducing weight representation by reducing bit width numbers, typically from floating point values to integer values. The most widely used quantization techniques are: i) post-training quantization, which envisages the model training followed by weight quantization, and as a last step a model (re)optimization to generate the quantized model; ii) quantization-aware training, which envisages the weight quantization during training, and then the network is re-trained to fine-tune the model precision to compensate the accuracy degradation occurred during the quantization process. Unfortunately, at present, it is not always clear how to balance the trade-off between compression, accuracy, and energy efficiency, and different techniques may perform differently for different models and for different DL tasks [2].

In this work, we carried out a thorough analysis of the energy effectiveness of the most recent quantization-based compression methods applied to a sensor-based Human Activity Recognition (HAR) case study resolved by deep learning models. HAR represents a set of tasks that gained a lot of attention in recent years because of the broad range of real-world applications. Early diagnosis, rehabilitation, and patient assistance can be provided in medical decision processes for healthcare monitoring purposes; industrial applications, gaming, and sport/fitness tracking are of great interest as well. Two main approaches are leveraged for HAR: camera-based and sensor-based recognition. Camera and inertial sensors allow to detect a set of daily human activities via computer vision techniques and acceleration/location sensors, respectively.

In this work, we focus on real-time sensor-based HAR tasks directly executed on top of a low-power wearable device characterized by strong real resource constraints. The contribution of this paper can be summarized as follows:

– we provide a methodology to fine-tune models hyperparameters while taking into account the energy efficiency applied to convolutional neural networks (CNN) and Long short-term memory (LSTM) neural networks.
– we characterize the accuracy-energy trade-off of different quantization-based compression techniques
– we made real energy consumption measurements in a real case study scenario by porting and executing each model on top of a real low-power wearable hardware setup.

## 2 Related Work

In recent years, researchers have started exploring the interplay between DL model compression techniques and energy efficiency. This includes developing compression techniques that explicitly consider energy consumption as a metric, as well as developing hardware-aware compression techniques that optimize the compressed model's energy consumption on specific hardware platforms [3].

The investigation of HAR on low-power microcontroller units (MCUs) blossomed in the last few years: back in 2020, Novac et al. evaluated the implementation of multi-layer perceptrons and convolutional neural networks for HAR on an ARM-Cortex-M4F-based MCU [15]. They compared the supervised learning methods to the unsupervised and online learning ones, by proving the higher benefits of the latter. Authors refer to online learning as the ability of a neural network to adapt itself to a new set of data, even though the initial learning phase is over. They further explored the deployment of HAR on MCU by proposing a new quantization method, together with a new framework that allows training, quantizing, and deploying deep neural networks [14]. In their work, they only consider convolutional neural networks (specifically, the ResNetv1 model architecture); int8, int16, and float32 were considered as quantization techniques, and SparkFun Edge and Nucleo-L452RE-P were considered as MCU platforms (both belonging to the Cortex-M4F core family).

Daghero et al. applied Binary Neural Networks (BNNs) to HAR to decrease network complexity via an extreme form of quantization [6]; indeed, by using BNNs the precision of data format, both weights and layers input/output, is reduced to 1-bit precision. Authors propose a BNN inference library that targets RISC-V processors.Subsequently, authors extended their work [4, 5] by proposing a set of efficient one-dimensional convolutional neural networks (1D CNNs) and testing optimization techniques such as sub-type and mixed-precision quantization. The aim was to find a good trade-off between accuracy and memory occupation. As a target platform, they leveraged again the RISC-V MCU.

Similarly, Ghibellini et al. proposed a CNN model for falling and running detection within an industrial environment for safety purposes [8]. This work shows very preliminary results in terms of accuracy and model size. Dynamic range quantization was applied to reduce the size of the model which is then deployed on the firmware of an Arduino BLE 33 Sense.

Compared to the existing literature, besides the one-dimensional convolutional neural network, which is the only one considered in all the others works,

we explore also the deployment, on a target MCU, of another deep neural network (DNN) relevant to the context of HAR. We consider the combination of the one-dimensional convolutional neural network (1D_CNN) with the vanilla long short-term memory (LSTM). The goal of this work is twofold: on the one hand, we want to perform an in-depth analysis that compares the deployment of relevant DNNs on a target MCU; on the other one, we want to provide insights on: i) the feasibility of the deployment of those networks on the target MCU; ii) a comparison in terms of classification performance and power-efficiency.

## 3 Methodology

In this section, we describe the proposed methodology to characterize in terms of classification accuracy and energy consumption, the state-of-the-art compression techniques applied to two types of deep neural networks. In particular, we considered the one-dimension convolutional network (1D_CNN) and the combined adoption of CNN and LSTM (1D_CNN_LSTM) applied to a HAR case study deployed on an Espressif ESP32-wroom-32 DevKit representing a real resource-constrained wearable device [7]. The ESP32 has two CPU cores that can be individually controlled running at a clock frequency adjustable from 80 MHz to 240 MHz. The chip also has a low-power mode that can be used to save power while performing peripheral I/O tasks that do not require much computing effort. Our version of the ESP32 has 4 MB of flash memory for saving the firmware and 400 KB of RAM memory.

Then we focused on tuning those hyperparameters that impact the complexity, the memory footprint, and the effectiveness of the network. On top of the obtained models, we applied compression techniques to investigate the tradeoff between accuracy loss and energy saving.

### 3.1 Hyperparameters tuning

The proposed 1D_CNN network envisages one convolutional 1D layer, followed by a pooling layer and two fully connected dense layers, in which the last one provides the output. It is worth mentioning that: i) a dropout layer is placed after the 1D convolutional layer; ii) a flatten layer is placed after the pooling layer; iii) the softmax function is used in the last layer to infer the activity label. For what concern the combined approach 1D_CNN_LSTM, the network structure is as follows: one convolutional 1D layer followed by a batch normalization and by a ReLU layer, which is then followed by a LSTM layer, followed by a dropout layer and two fully connected dense layers, in which the last one provides the output.

Table 1 lists the network types highlighting the hyperparameters adopted in our investigation. The network ID is used as a shortcut to refer to the network type and its relative structure. The network structure labels are coded by using the $xxT$ notation where $xx$ is the number of units and $T$ represents the type of unit. For instance, in the case of C_8, the string *32F 3K 100D* stays for 32

convolutional filters with a kernel size of 3 followed by a dense layer containing 100 neurons.

**Table 1.** Network types and structure.

| Net Type | Net ID | Net Structure | Net Type | Net ID | Net Structure |
|---|---|---|---|---|---|
| 1D_CNN | C_8 | 32F 3K 100D | 1D_CNN_LSTM | – | – |
| | C_7 | 20F 3K 100D | | – | – |
| | C_6 | 16F 3K 100D | | CL_6 | 16F 3K 16H 256D |
| | C_5 | 10F 3K 100D | | CL_5 | 16F 3K 8H 256D |
| | C_4 | 8F 3K 100D | | CL_4 | 16F 3K 4H 256D |
| | C_3 | 4F 3K 100D | | CL_3 | 8F 3K 16H 256D |
| | C_2 | 2F 3K 100D | | CL_2 | 4F 3K 16H 256D |
| | C_1 | 1F 3K 100D | | CL_1 | 2F 3K 16H 256D |

The hyperparameter tuning in deep-learning models dedicated to a tiny device must be carefully conducted because of its resource limitations. In particular, the network dimension must be compatible with the device memory characteristics of the device, together with its execution time (i.e. the time needed to make an inference). After a brief manual tuning of the structure parameters, we end up in the network structures listed in Table 1. For each network type, we decide to start from what we expect to be the biggest size capable of fitting inside the device, and from that size, we then decrease one, or two, parameters in order to lower the network complexity.

For (1D_CNN) we made use of 3 kernels because we have seen it is a value typically used in literature; instead, the choice of the filter value is related to the fine-tuning phase. To gradually decrease the network complexity we decide to reduce the number of filters in the convolutional layers while keeping the same number of kernels, planning a total of 8 network models. For the combined network (1D_CNN_LSTM), starting from the bigger network installable in the device memory, we planned 6 network models obtained by reducing the number of filters in the convolutional layer.

### 3.2 Models characterization workflow

The proposed characterization workflow consists of three phases highlighted in Fig. 1. In the first phase, the raw data sampled from a 3-axial accelerometer and gyroscope are extracted from a dataset and are fed into the base network model to perform train and subsequent evaluation. The raw signals have been divided into time windows to generate samples for training and testing purposes. In the second phase, the compression techniques have been applied to the trained models. In particular, first of all, we applied the lite conversion to the trained model, then the post-training dynamic quantization, and the post-training full-integer quantization.

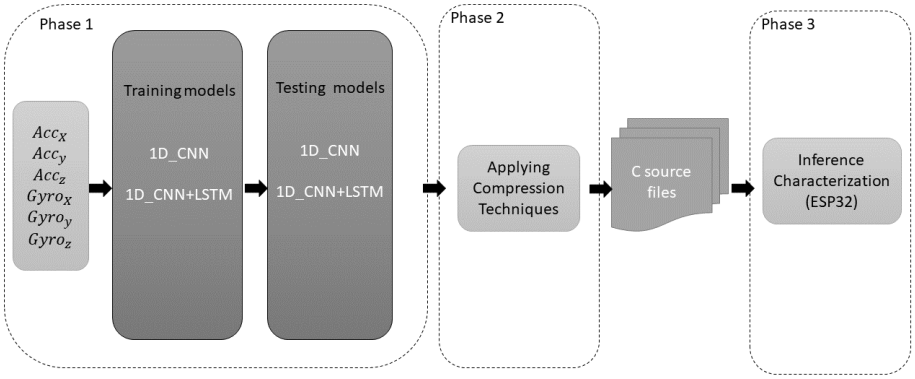



**Fig. 1.** General workflow.

The application of each compression technique via the TensorFlow lite converter allows us to generate a C source file containing the char array version of the lite. In the third phase, each C source file is compiled into an executable file by means of the ESP-IDF platform (Espressif IoT Development Framework) and moved onto the ESP32 device. Here the execution time and energy have been measured. It is worth mentioning that the proposed workflow has been applied to each network listed in Table 1.

## 4 Experimental Evaluation

All the deep learning models have been trained and tested using a Keras-TensorFlow application running in a Google Colab environment [16]. TF provides also TensorFlow Lite(TFL) a library for network model deployment on mobile devices, microcontrollers, and edge devices. In particular, TFL allows converting a base TF model into a compressed version via the so-called TFLite converter by applying different compression techniques such as post-training dynamic quantization and full-integer 8-bit quantization.

As a case study we developed a wearable sensor-based HAR application on top of an Espressif ESP32-wroom-32 DevKit [7] connected to an MPU6050 integrated 6-axis motion tracking device that combines a 3-axis gyroscope, and 3-axis accelerometer [10]. The application was entirely developed in C++ using the ESP-IDF platform with the TensorFlow Lite Micro libraries installed.

To train the models and test their accuracy we leveraged the UCI-HAR dataset [17] which is a publicly available dataset where accelerometer and gyroscope signals are sampled at 50 Hz. The monitored activities are gathered from

30 subjects aged between 18 and 48 years and are: *walking, walking upstairs, walking downstairs, sitting, standing, and lying down.* In this work, the entire dataset was split into 75% for training and 25% for testing.

To estimate the energy consumption of the ESP32 device, we measured the voltage drop across a sensing resistor ($9.8\Omega$) placed in series with the device's power supply. The device was powered at 3.3V through an NGMO2 Rohde & Schwarz dual-channel power supply [18], and we sampled the signals to be monitored during the experiments by means of a National Instruments NI-DAQmx PCI-6251 16-channel data acquisition board [13].

### 4.1 Results

This section reports the performance characterization of the compression techniques applied to the proposed deep-learning models. In particular, we provide a comparison in terms of classification accuracy and energy consumption.

**Classification accuracy** Table 2 reports the highest classification accuracy reached by each network type before (base) and after the application of the compression techniques.

**Table 2.** Highest classification accuracy reached by each network type considering the application of each compression technique. The "base" case refers to a model not subject to any compression techniques.

| network | | accuracy | | | |
|---|---|---|---|---|---|
| type | configuration | base | lite | dynamic | full-integer |
| 1D_CNN | C_3 | 0.887 | 0.884 | 0.888 | 0.858 |
| 1D_CNN_LSTM | CL_3 | 0.918 | 0.917 | 0.922 | 0.902 |

These results, first of all, highlight the already known supremacy of the combined network with respect to the 1D_CNN which scores nearly 3 percentage points more accuracy in the uncompressed configuration. On the other hand, going from the basic model to the lite one, no significant loss of accuracy is found and, even if dynamic compression is applied to the latter, the classification performance still remains the same. Notice that, the negligible increase in accuracy, in the latter case, is due to the different routines used in TensorFlow to characterize lite models with respect to base models. Applying the full-integer quantization to both models, on the other hand, a loss of accuracy ranging from 2 to 3 percentage points is obtained.

Table 3 reports with an ✗all the network models with respect to the compression technique: Lite (L), Dynamic (D), and Full-integer (F) not fitting inside the device. It is interesting to note that by simply lite converting the 1D_CNN_LSTM we never obtain a fitting and running network model.

**Table 3.** Network models fitting on the wearable device.

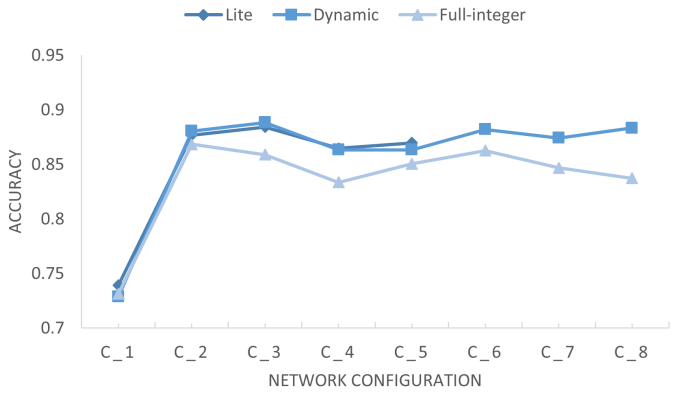
| Net Type | Net ID | L | D | F | Net Type | Net ID | L | D | F |
|---|---|---|---|---|---|---|---|---|---|
| 1D_CNN | C_8 | ✗ | ✓ | ✓ | 1D_CNN_LSTM | – | – | – | – |
| | C_7 | ✗ | ✓ | ✓ | | – | – | – | – |
| | C_6 | ✗ | ✓ | ✓ | | CL_6 | ✗ | ✓ | ✓ |
| | C_5 | ✓ | ✓ | ✓ | | CL_5 | ✗ | ✓ | ✓ |
| | C_4 | ✓ | ✓ | ✓ | | CL_4 | ✗ | ✓ | ✓ |
| | C_3 | ✓ | ✓ | ✓ | | CL_3 | ✗ | ✓ | ✓ |
| | C_2 | ✓ | ✓ | ✓ | | CL_2 | ✗ | ✓ | ✓ |
| | C_1 | ✓ | ✓ | ✓ | | CL_1 | ✗ | ✓ | ✓ |

Figures 2 (a) and (b) show the accuracy provided by each compression technique applied respectively to the 1D_CNN and 1D_CNN_LSTM models when the network complexity is increased. Concerning the 1D_CNN network, the increase in complexity from C_1 to C_2 results in a large increase in the classification accuracy which reaches its maximum value with C_3 and then keeps it at high levels for both Lite and Dynamic models. Notice that, the accuracy of the configurations C_6, C_7, and C_8 of the Lite models are not plotted since they do not fit into the memory device. The significant loss of accuracy due to full-integer quantization is also evident from the graph for each network configuration.

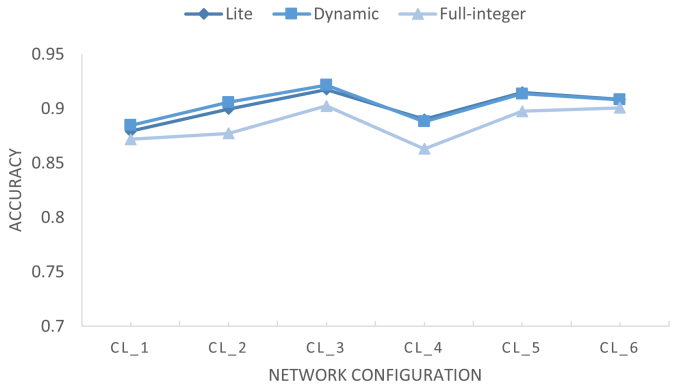
The behavior of the 1D_CNN_LSTM network is almost the same with an increase in accuracy for more complex configurations and a non-negligible accuracy loss in the case of the full-integer quantization.

**Energy consumption** For each network type, all the respective fitting network models reported in Table 3 were deployed one by one on the ESP32 device. Given a running model, the HAR application was run to sample real-time data directly from the gyroscope and the accelerometer and to perform real-time inference measurements. In particular, for each model, we collected 10 consecutive measures in order to get an average of the inference time. Moreover, during execution, the device was connected to the energy measurement setup to sample the corresponding current consumption waveforms form which to derive the inference energy of each model.

Figures 3 (a) and (b) show the inference energy measured respectively for the 1D_CNN and 1D_CNN_LSTM models when varying the hyperparameters configuration. In both cases, increasing the model complexity produces an increase in the energy needed to make an inference. In the 1D_CNN case also the energy consumed by the simple lite model for configurations C_1 to C_5 is reported highlighting a non-negligible energy saving produced by dynamic and full-integer quantization techniques with respect to the raw lite models. On the other hand, the full-integer quantization does not appear to introduce an additional energy advantage over dynamic quantization. Notice that, unfortunately, no configuration of lite models was able to run on the device in the case of the




(a) 1D_CNN

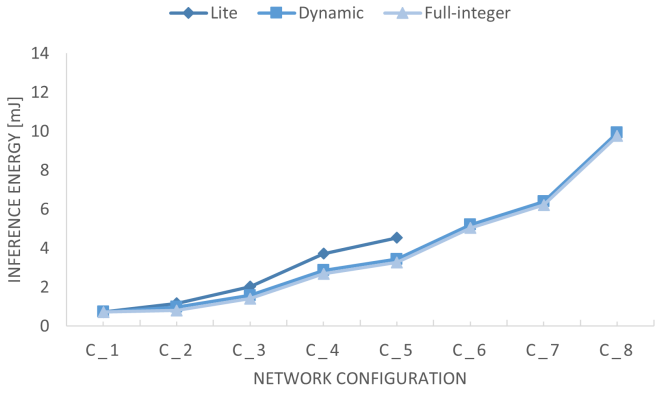



(b) 1D_CNN_LSTM

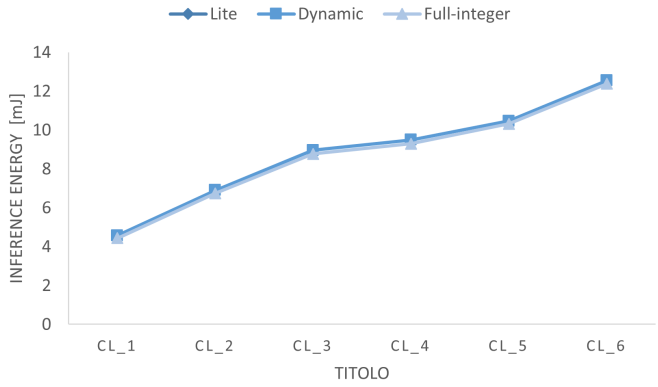
**Fig. 2.** Classification accuracy of each network type per increasing network complexity, obtained by applying lite conversion, dynamic quantization, and full-integer quantization

CNN and LSTM combined network so we do have not an energy consumption baseline in this case.

In the case of model 1D_CNN, for the configurations from C_1 to C_5, we have estimated the energy savings induced by the two quantization techniques with respect to the base value of the lite model. Figure 4 shows the corresponding bar graph. Interestingly, for the simplest model (C_1) both techniques do not affect the energy consumption which is always very low. Starting from C_2 to C_5 configurations, the energy saved by the full-integer quantization always overcomes that obtained with the dynamic technique even if, it seems that as the




(a) 1D_CNN




(b) 1D_CNN_LSTM

**Fig. 3.** Energy obtained by applying lite conversion, dynamic quantization and full-integer quantization

complexity of the models increases, the differences between the two techniques decrease, both reaching around an energy-saving close to 30%.

Considering that, from the classification accuracy point of view, the dynamic quantization technique does not involve a priceable loss, we would like to advise in any case the use of this technique even if the energy savings produced by the full-integer quantization could be slightly greater. Furthermore, we remind you that the application of one of the two quantization techniques is even indispensable for models of considerable size which otherwise would not execute on some extremely low-power platforms such as the ESP32.

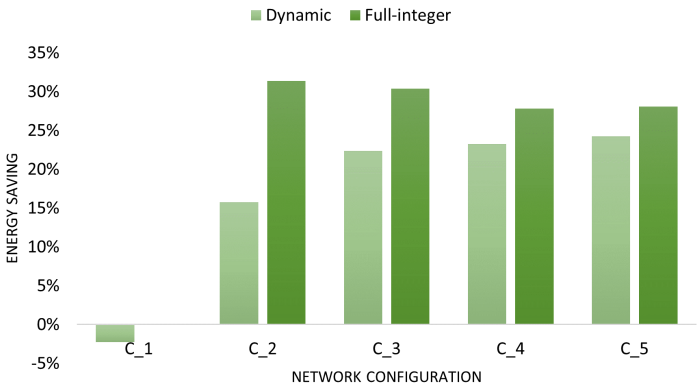



**Fig. 4.** Energy saving obtained by applying dynamic and full-integer quantization on the 1D_CNN model when varying its complexity.

## 5 Conclusions

In this paper, we carried out a thorough analysis of the energy effectiveness of the most recent quantization-based compression methods applied to a HAR case study resolved by deep learning models. We specifically looked at CNN models and CNN and LSTM combinations. The models were run on a real low-power wearable device while concurrently monitoring its energy consumption after being converted and compressed from a dataset that is widely used in the field of HAR applications.

From the classification point of view, the dynamic quantization technique and the lite conversion result in a negligible loss of accuracy while the full-integer recorded losses between 2% and 3%. On the other hand, for both dynamic and full-integer compression techniques, we measured an energy saving of close to 30%.

As a final remark, we would like to advise in any case the use of the dynamic quantization technique considering that, from the classification accuracy point of view, it does not involve a priceable loss. Furthermore, we confirm the need of applying one of these quantization techniques for several models of considerable size which otherwise would not execute on some extremely low-power platforms such as the ESP32.

## References

1. Augasta, M., Kathirvalavakumar, T.: Pruning algorithms of neural networksâĂŤa comparative study. Open Computer Science **3**(3), 105–115 (2013)
2. Chaman, S.: Techniques for compressing deep convolutional neural network. In: 2020 International Conference on Computational Performance Evaluation (ComPE). pp. 048–053. IEEE (2020)

3. Choudhary, T., Mishra, V., Goswami, A., Sarangapani, J.: A comprehensive survey on model compression and acceleration. Artificial Intelligence Review **53**, 5113–5155 (2020)
4. Daghero, F., Burrello, A., Xie, C., Castellano, M., Gandolfi, L., Calimera, A., Macii, E., Poncino, M., Pagliari, D.J.: Human activity recognition on microcontrollers with quantized and adaptive deep neural networks. ACM Transactions on Embedded Computing Systems (TECS) **21**(4), 1–28 (2022)
5. Daghero, F., Pagliari, D.J., Poncino, M.: Two-stage human activity recognition on microcontrollers with decision trees and cnns. In: 2022 17th Conference on Ph. D Research in Microelectronics and Electronics (PRIME). pp. 173–176. IEEE (2022)
6. Daghero, F., Xie, C., Pagliari, D.J., Burrello, A., Castellano, M., Gandolfi, L., Calimera, A., Macii, E., Poncino, M.: Ultra-compact binary neural networks for human activity recognition on risc-v processors. In: Proceedings of the 18th ACM International Conference on Computing Frontiers. pp. 3–11 (2021)
7. Espressif: Esp32-c3-wroom-02 datasheet (2022), https://www.espressif.com/en/support/documents/technical-documents, last accessed 2023-02-07
8. Ghibellini, A., Bononi, L., Di Felice, M.: Intelligence at the iot edge: activity recognition with low-power microcontrollers and convolutional neural networks. In: 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC). pp. 707–710. IEEE (2022)
9. Gou, J., Yu, B., Maybank, S.J., Tao, D.: Knowledge distillation: A survey. International Journal of Computer Vision **129**, 1789–1819 (2021)
10. InvenSense Inc.: Mpu-6050 product specification (2023), https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/, last accessed 2023-02-07
11. Khoram, S., Li, J.: Adaptive quantization of neural networks. In: International Conference on Learning Representations (2018)
12. Liang, T., Glossner, J., Wang, L., Shi, S., Zhang, X.: Pruning and quantization for deep neural network acceleration: A survey. Neurocomputing **461**, 370–403 (2021)
13. National.Instruments: Pc-6251 datasheet (2020), http://www.ni.com/pdf/manuals/375213c.pdf, last accessed 2023-02-07
14. Novac, P.E., Boukli Hacene, G., Pegatoquet, A., Miramond, B., Gripon, V.: Quantization and deployment of deep neural networks on microcontrollers. Sensors **21**(9), 2984 (2021)
15. Novac, P.E., Castagnetti, A., Russo, A., Miramond, B., Pegatoquet, A., Verdier, F.: Toward unsupervised human activity recognition on microcontroller units. In: 2020 23rd Euromicro Conference on Digital System Design (DSD). pp. 542–550. IEEE (2020)
16. Pang, B., Nijkamp, E., Wu, Y.N.: Deep learning with tensorflow: A review. Journal of Educational and Behavioral Statistics **45**(2), 227–248 (2020)
17. Reyes-Ortiz, J.L., Oneto, L., Samà, A., Parra, X., Anguita, D.: Transition-aware human activity recognition using smartphones. Neurocomputing **171**, 754–767 (2016)
18. Rohde&Schwarz: Ngmo2 datasheet (2020), https://www.rohde-schwarz.com/it/brochure-scheda-tecnica/ngmo2/, last accessed 2023-02-07