Optimizing Computational Resources for Edge Intelligence Through Model Cascade Strategies

Oihane Gómez-Carmona, Diego Casado-Mansilla, Diego López-de-Ipiña, *Member, IEEE*, and Javier García-Zubia, *Senior Member, IEEE*

***Abstract*—As the number of interconnected devices increases and more artificial intelligence (AI) applications upon the Internet of Things (IoT) start to flourish, so does the environmental cost of the computational resources needed to send and process all the generated data. Therefore, promoting the optimization of AI applications is a key factor for the sustainable development of IoT solutions. Paradigms such as Edge Computing are progressively proposed as a solution in the IoT field, becoming an alternative to delegate all the computation to the Cloud. However, bringing the computation to the local stage is limited by the resources' availability of the devices hosted at the Edge of the network. For this reason, this work presents an approach that simplifies the complexity of supervised learning algorithms at the Edge. Specifically, it separates complex models into multiple simpler classifiers forming a cascade of discriminative models. The suitability of this proposal in a human activity recognition (HAR) context is assessed by comparing the performance of three different variations of this strategy. Furthermore, its computational cost is analyzed in several resource-constrained Edge devices in terms of processing time. The experimental results show the viability of this approach to outperform other ensemble methods, i.e., the Stacking technique. Moreover, it substantially reduces the computational cost of the classification tasks by more than 60% without a significant accuracy loss (around 3.5%). This highlights the potential of this strategy to reduce resource and energy requirements in IoT architectures and promote more efficient and sustainable classification solutions.**

***Index Terms*—Edge computing, edge intelligence, embedded systems, ensemble learning, Internet of Things (IoT), machine learning (ML), optimization.**

I. Introduction

ALONG with the rapid emergence of the Internet of Things (IoT) and its progressively wider impact on many sectors, the number of interconnected devices has increased exponentially [1]. At the same time, the growing demand for data from artificial intelligence (AI) applications motivates the interest in monitoring and capturing as much information as possible. However, while connecting large numbers of low-resource and low-cost nodes is more affordable than ever, the environmental costs of transmitting and processing the amount of data they generate are escalating at a fast rate [2].

IoT ecosystems have traditionally relied on remote Cloud data centers to carry out the fundamental parts of the data analysis. This means that, despite being obtained locally, the information needs to travel long distances and reach remote data centers where the processing capabilities are placed [3]. The volume of information produced by IoT devices every day makes transmitting every byte not favorable in terms of both resources and time. In consequence, this emphasizes the scalability and sustainability issues of traditional Cloud-based scenarios and brings into question the high energy and computational costs associated with the data-driven interconnected future that IoT envisages [4].

To solve the latter issue, it becomes more logical to place at the Edge as much analytical power as possible since data is generated at sensors close to the users, factories, or cities. For this reason, the current body of the research proposes a paradigm shift from the legacy systems, that have traditionally relied on remote Cloud data centers, to the concept of Edge Intelligence, that pushes the decision-making process to local devices [5]. However, this transition comes at a price. In this case, the Edge paradigm poses a considerable challenge in terms of the computational and storage capabilities needed for the devices hosted at the Edge. As processing techniques and machine learning (ML) applications are resource-demanding tasks, alleviating their computational load becomes necessary to undertake heavy operations in such constrained settings.

In consequence, several research efforts have been made to design more efficient Edge solutions. In this direction, some works delve into the possibility of sharing heterogeneous computation and communication resources among devices and creating new offloading schemes for a decentralized Cloud [6] or Mobile Edge Computation [7], and improving the resource management of Federated Learning approaches [8]. Other works address the idea of embedding the processing capabilities on the devices themselves and propose running AI models solely on the device [9]. The first approach copes with the requirements of running computation-intensive algorithms by proposing a synergy between Edge devices and even Edge-Cloud approaches.

Manuscript received July 19, 2021; revised September 10, 2021; accepted October 2, 2021. Date of publication October 8, 2021; date of current version May 9, 2022. This work was supported in part by the European Commission through the AURORAL Project under Grant 101016854, and in part by the Ministry of Economy, Industry and Competitiveness of Spain for IoP, under Grant PID2020-119682RB-I00. *(Corresponding author: Oihane Gómez-Carmona.)*

Oihane Gómez-Carmona, Diego Casado-Mansilla, and Diego López-de-Ipiña are with the DeustoTech, University of Deusto, 48007 Bilbao, Spain (e-mail: oihane.gomezc@deusto.es; dcasado@deusto.es; dipina@deusto.es).

Javier García-Zubia is with the Faculty of Engineering, University of Deusto, 48007 Bilbao, Spain (e-mail: zubia@deusto.es).

Digital Object Identifier 10.1109/JIOT.2021.3118845

In contrast, the second one reduces the path length of data offloading and favors data privacy at the local stage [10]. Thus, one of the main challenges of ML techniques being executed upon IoT is related to improving the efficiency of Edge intelligent systems for different application scenarios [11]. As so, in both approaches, the tradeoff between optimality and efficiency should be studied to balance the computational cost and the accuracy of the system based on the characteristics of the network infrastructure. Thus, saving time and resources while avoiding as much externalization of data as possible [12].

To contribute to the sustainability of IoT systems and Edge Intelligence solutions, in this article we evaluate a multistage ML strategy to ease the optimization of the data processing power in Edge nodes. In particular, we implement an adaptive ensemble learning approach through a discriminative model cascade, aiming at matching the computational needs of classification tasks to their complexity. Here, complex ML systems are divided into successive layers of simpler models that filter input elements at each stage of the cascade based on the confidence of the prediction at that level. In this approach, the simplicity of the models depends on the number of features that are considered at each level. This number of features increases as the cascade progresses. Thus, optimization comes from having simple classifiers at the initial levels of the cascade to increase the efficiency of the system when classifying the easiest instances. Then, more complex classifiers at the final levels of the cascade are considered for a refinement of the classification accuracy in those hardest to classify instances. This reduces the overall computational time and avoids oversizing the computational resources dedicated to each classification task. Thus, the proposed strategy contributes to mitigating the environmental impact of such solutions proposing a more efficient and dynamic usage of the available resources of each node.

To evaluate its suitability in Edge settings, we present three novel variations of the training process of this discriminative cascade of models. Then, we assess them in a human activity recognition (HAR) context. Their different results are analyzed by providing the tradeoff between efficiency and classification results. To this aim, four publicly available HAR data sets and a novel data set produced for this research, which is related to office hydration habits recognition, are used. Finally, we empirically analyze the performance, in terms of classification time, of a scenario resembling a real-world IoT system. We perform this evaluation on the introduced novel data set in several low-cost Edge devices (namely, Nvidia Jetson Nano, Raspberry Pi 3, Raspberry Pi 4, and Raspberry Pi Zero). The results show the potential of the proposed strategy to simplify the classification tasks in constrained settings (Edge/users' devices) while outperforming Stacking techniques and maintaining similar classification results as the reference models.

In essence, the contributions of this article are twofold.

1) An ensemble learning approach is proposed, with three novel variations, to increase the efficiency of classification tasks through a discriminative model cascade.
2) An evaluation of its potential to reduce computational time is presented, aiming to optimize the available resources of Edge devices and to improve the sustainability of ML-based IoT systems.

The remainder of this article is organized as follows. Section II outlines the literature on ensemble learning and optimization techniques. Section III describes the presented approach. Section IV explains the procedure and the experimental setup. The obtained results are presented in Section V and discussed in Section VI, with a special emphasis on the relevance of energy-efficient AI systems. Finally, Section VII draws some conclusions and outlines the future work.

II. Related Work

Ensemble learning techniques have been widely covered in ML research. Approaches like Bagging [13], Boosting [14], or Stacking [15] are well-known methods that have traditionally been used to improve the general performance of classification systems by combining several base models in a final predictive meta-model [16]. Cascading is another particular example of ensemble learning. First introduced by Viola and Jones [17], it consists of the concatenation of several classifiers to minimize the generalization error rate. HAR applications [18] or the classification of rare events in face detection [19] are examples of domains in which cascade ensemble learning is usually applied.

Even though these strategies improve the predictive performance of a single model, they also tend to increase the computational complexity of training the ensemble and predicting new instances [20]. An interesting alternative is to use the combination of models of cascading approaches in a cost-sensitive manner. In this regard, the complexity of the ensemble methods is faced by implementing several increasingly complex classifiers that divide the computation into different stages and classify the input data with the stage that fits the most to its difficulty [21]. This idea was first introduced by Kaynak and Alpaydin, who proposed a multistage cascading scheme that relies on the concatenation of several small classifiers [22]. The objective of this methodology was to reduce computational cost yet not losing much in terms of accuracy. The authors assessed their proposal in a two-stage approach. The former classifier was based on single-layer and multilayer perceptron (MLP) models, considered lighter prediction methods. The latter classifier was based on k-nearest neighbors (KNN). To improve the classification results, successive models were trained with a self-modified training set, prioritizing the uncovered patterns or data that the previous models overlooked. Additionally, confidence thresholds to define if the classification was acceptable were defined based on previous probabilities rates.

Silva *et al.* [23] proposed a classification approach also based on a two-stage combination of monolithic and ensemble classifiers. The rationale behind their proposal was to deal with the majority of unclassified instances using the monolithic classifier leaving the most challenging instances of classification problems for the ensemble classifier systems. The latter classifier was based on pool generation methods. This way, in this work, the complexity of every stage of the system is determined by the inherent complexity of the generation methods.

In particular, they applied KNN and support vector machines (SVMs) methods for the initial model and different ensemble techniques (i.e., bagging, boosting, and random subspace selection—RSS) for the final model. Since the authors compared their approach against other multiple classifiers systems, it remains an open question whether such a method outperforms a single monolithic classifier in optimization. Other approaches have proposed additional simplification techniques by understanding the cascading techniques as a tree topology case [24], [25]. In this case, different topologies can be combined to construct a tree of classifiers that determine how many features are needed to predict new unseen data.

The reviewed methods for optimizing the classification of data are often done at the cost of losing accuracy. This highlights the need for additional adjustments to compensate for this loss. As introduced above, Kaynak and Alpaydin [22] coped with accuracy loss by specifically tuning each of the successive models and thresholds to better match with previously uncovered patterns. Other authors usually require to consider the influence of additional/side factors, such as the rejection margin and fit the models accordingly [26]. In this case, to keep the loss in accuracy to a minimum, it is important to analyze the reasons for accepting or rejecting the instances classified by each of the models that comprise the cascade. Oliveira *et al.* [27] studied the optimum class-related reject threshold to provide a better error rejection in cascading classifiers. The proposed methodology proved to improve the performance of a system based on a set of MLP models for handwritten digits recognition systems. Zhang *et al.* [28] increased the recognition reliability by using a double-check mechanism to weight and verify the neural network (NN) confidence values and to correct the errors. Nevertheless, the computational cost was not considered in their work.

Regarding NN, Wang *et al.* [29] analyzed how cascade models could outperform in efficiency other ensemble-based architectures. Similarly, adapting the size and the number of layers of those models has been a subject of deep research. One approach is defining a combination of big/little models [30], or cascading the depth of the layers in NN techniques [31]. Similarly, Taylor *et al.* [32] faced the intrinsic complexity of the Deep Learning models by presenting an adaptive scheme to determine which model to use for a given input. Considering the desired accuracy and inference time, their approach employed a KNN predictive model to select the pretrained models that best suited a given input and the optimization constraint. In parallel, other approaches leave open the possibility to adapt their work in an optimized way [33].

Going in a different direction with respect to the reviewed body of knowledge, this work targets IoT architectures and Edge Computing approaches. Therefore, it aims to shed light on the actual optimization opportunities a discriminative cascade of models has in constrained settings, providing an empirical demonstration of its potential computational cost-savings in real Edge devices. For that, we present and evaluate three novel variations for the training setup of the cascade strategy. In all cases, the cascade method's complexity is given by the number of characteristics of the input data. This way, we evaluate the influence of the simplicity of the models in terms of feature computation since the computation time required for feature extraction can penalize those applications that rely on the characteristics obtained from the signal [34]. This optimizes both the prediction task and the data processing stage at each level of the cascade, contributing to the efficiency of the cascade in a greater way than basing the definition of its levels around the completeness of the classification algorithm. In this regard, the potential accuracy loss of relying on a reduced subset of features is faced by setting up confidence thresholds that are more restrictive for those potentially weaker cascade stages. Additionally, we also evaluate the suitability of some of the most common classic ML algorithms to be part of a model cascade, complementing previous studies that are mainly focused on specific cascade configurations. In the following, we will present our approach and examine its suitability for HAR contexts.

III. Model Cascade Optimization Approach

In this section, we describe a particular ensemble learning method for constrained contexts. In this work, this method is defined as an optimization mechanism. Given a particular input (e.g., a new data stream captured by an IoT sensor), it faces the complexity of the classification techniques by adapting the resources of the node in charge of the prediction process to the difficulty of the task. We also introduce three variations of the training process that seek to find the most suitable strategy to distribute the discovery of uncovered patterns across the included models.

A. Proposed Discriminative Cascade for Inference

The presented cascade ensemble method is comprised of a series of successive N discriminative models, starting from Model 1. Based on the confidence of the prediction at each level i, a particular data input may requiere a more fine-grained prediction with the following model at level $i + 1$, which is more complex in terms of features, given some confidence thresholds t (hyperparameter to be tuned). If the prediction of a sample at level i is considered to be reliable, the sample is removed from the pipeline and classified using ith model's prediction. If none of the different $N - 1$ models is able to provide a reliable prediction within these thresholds t_i, the final Nth model would be in charge of classifying the remaining instances. Notice that the latter model is created using all the available features so that it has the maximum amount of information that is possible for the task. A schematic representation of this system is presented in Fig. 1. This way, simpler models act as filters that make it less likely for the data to reach the complex ones, allowing for a reduced computational burden and time to process the samples through the $N-i$ models. Therefore, the complexity of the final application is reduced by choosing from an initial cascade of pretrained models the configuration that fits the most to the complexity of the prediction.

The confidence values are determined by the prediction probabilities, which summarize the likelihood (or uncertainty) of samples belonging to each of the available classes (in our

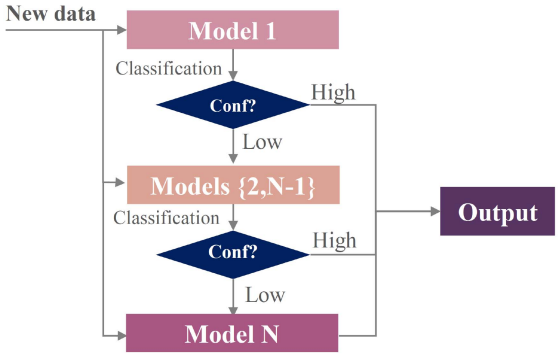

Fig. 1. Schematic representation of the workflow for classifying new samples.

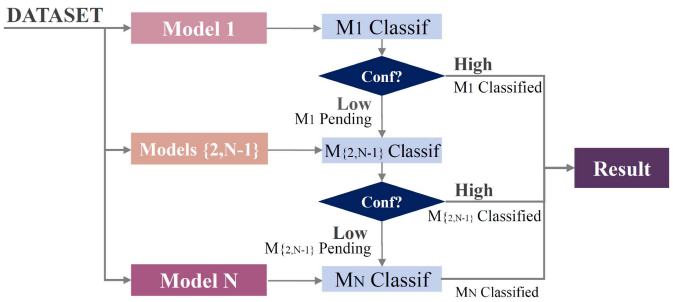

Fig. 2. Schematic representation of the workflow for parallel training.

case, human activities). In this context, prediction probabilities represent the confidence a model has in the predictions, i.e., in samples belonging to the different classes. If the confidence of a sample being of class c is high, the model is confident the sample should be classified as class c. In this specific proposal, this value is calculated through a multiclass log loss metric (also referred as categorical cross-entropy).

This metric calculates the negative log-likelihood for the probability predictions made by the classification model. Log loss values are always positive, being bounded between 0 and ∞. Thus, a log loss of 0 represents the minimum divergence between the prediction probability and its corresponding true value. In other words, the model can be considered to be entirely confident in the prediction. In this approach, instead of using log loss for evaluating the predicted probabilities and assessing the performance of a classification problem, we use this metric as an estimator of the reliability of a prediction in which the actual class is unknown. Hence, we calculate the multiclass log loss to obtain the loss coefficient by comparing the expected probability vector, constituted by the predicted probabilities of all the classes, against the predicted label. The multiclass log loss for each observation would be given by (1), which calculates the loss for each class label per observation and then sums the result

$$L(p, y_{\text{pred}}) = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{1}$$

where M is the number of classes, y_{pred} is a discrete binary value (0 or 1) of the predicted class c of the observation o, and p is the predicted probability of that observation belonging to class c.

This $L(p, y)$ value is compared against the numeric threshold set at each level; if the confidence $L(p, y)_i$ at stage i is above a specific threshold value t_i, the prediction is marked as valid. In this case, the lower the threshold, the more stringent the confidence requirements of the model, i.e., it is easier that a sample has to go through more stages until it is successfully classified with a confidence higher than t_i. From the optimization point, we avoid specific samples to be processed by more complex models than what is required for classifying that difficulty. Besides, the data processing stages of the signals and the feature extraction phases are improved. That is, just the features needed for each of the levels are computed, and the remaining features are only computed at the following levels of the cascade if required. Recall that the presented cascade approach may have all the new samples labeled with the initial and simpler classifiers without requiring the computation of all the features and just a subset of them.

B. Different Training Methods

To better understand the flexibility of the proposal, we have defined three different implementations of the system's training strategy. The difference among those implementations is determined by the input data used to train each of the layers. Resembling model ensemble techniques, we can differentiate those implementations in which all the models are trained with all the available data (parallel implementation) from the ones in which the successive layers of models are trained only with previously unclassified data (sequential implementation). We also present a mixture of both methods that seeks to combine the potential advantages of both approaches (hybrid implementation).

1) Parallel Implementation: In the parallel training all the models are trained using all the available data to, presumably, be able to generalize better on unseen samples. Therefore, all the levels are independent in terms of input. A schematic representation of the training set up is shown in Fig. 2. This training approach is similar to the one followed by the stacking model ensembling technique [15], in which several heterogeneous models learn in parallel with the whole training data set. However, contrary to the stacking method, in the cascade of models, there is no final stage of creating a meta-predictor or meta-model that combines the output of the different trained models.

2) Sequential Implementation: The second implementation uses the whole data set to train the initial model. However, the successive models are trained using only a fraction of this initial data set. At stage i, this fraction consists of those instances that do not have a reliability/confidence higher than t_i in the previous phase $i-1$. That is, only those instances that fail to pass all of these $N-1$ confidence tests are used in the construction of the following exception learner, as shown in Fig. 3. This way, the data input of the successive models is conditioned by the previous predictions. This allows having $N-1$ models forced to learn suitable patterns for hard to classify samples, covering the flaws of the initial model. Thus, it is designed to better discover uncovered patterns of the data. This strategy follows a similar approach to the Boosting model assembling

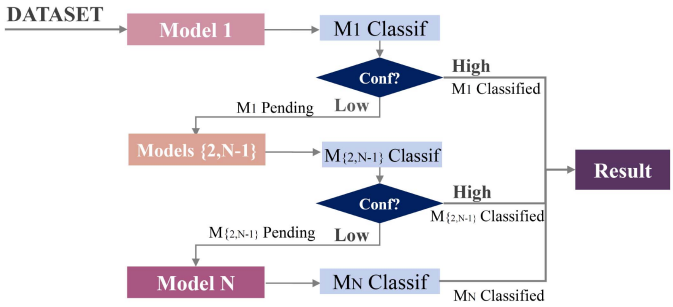

Fig. 3. Schematic representation of the workflow for sequential training.

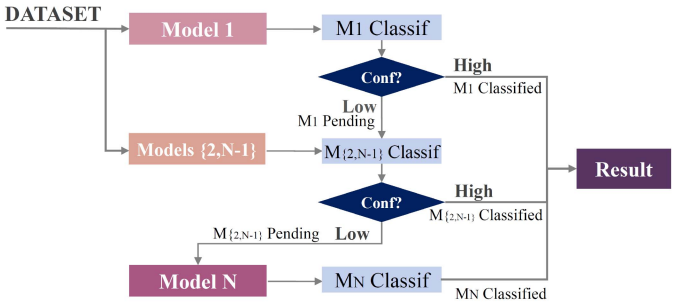

Fig. 4. Schematic representation of the workflow for hybrid training.

technique [14], where the successive homogeneous models are created based on the errors of the past ones and random sampling techniques. In contrast to this, in the cascade approach, new data does not need to go through all the trained models to be classified.

3) Hybrid Implementation: It presents a mixed approach in which the idea of parallel training is followed in all the stages except in the last stage of the cascade, as Fig. 4 illustrates. The objective of this implementation is to balance the generalization of the initial cascade stages (trained with all the available input data) with a greater particularization in the final model (trained only with only a fraction of this input data). This way, the last model is intended to specialize only in those patterns that have been left uncovered by the rest of the $N-1$ models and it is forced to be trained with few but hard to classify examples.

IV. Procedure and Methodology

In this section, we propose the use case of a HAR problem to analyze the suitability of the introduced approach for continuous time-series data (e.g., data from accelerometers). For this reason, by means of several public HAR data sets, we evaluate the performance of the different variations presented in Section III. Then, we analyze the suitability of the selected strategy when a single constrained IoT device works as the central element of an Edge architecture and is in charge of classifying new inputs of data.

A. Selected Data Sets

We have selected four publicly available data sets from the body of literature. These data sets aim to cover different contexts in HAR with data captured by motion sensors. Those data sets are: 1) ADL recognition with wrist-worn accelerometer data set [35]; 2) HAR using smartphones data set [36]; 3) office activity recognition using accelerometers and gyroscopes located on the forearms [37]; and 4) the daily and sports activities data set [38]. The latter data set is included in the ones recommended for elaborating benchmarks and comparatives of the performance of Edge devices for classification purposes [39]. In our case, only the torso and right arm sensors have been considered.

TABLE I
Main Characteristics of the Selected Data Sets

Dataset	Nº Instances	Nº Classes	Nº Initial Feat.
OHM Dataset [40]	1000	3	486
HAR Dataset [36]	10299	6	561
ADL dataset [35]	839	13	162
Office Dataset [37]	7551	5	272
Sports Dataset [38]	9120	19	162

Besides, we also present a novel data set created explicitly for this research, namely, the office hydration monitoring (OHM) Data set. It focuses on classifying office employees' hydration patterns based on an accelerometer and a gyroscope sensor placed on different liquid containers (e.g., mugs, bottles, or glasses). It contains 1000 instances performed by ten subjects and includes 25 variations of different interactions that could be made with liquid containers. Those interactions are grouped into three main classes: 1) drinking from a bottle; 2) drinking from a cup; and 3) other kinds of interactions). This data set was created with the idea of having a noncontrolled activity data set that resembles real-world scenarios. Therefore, the interaction to be recorded was intentionally described very vaguely to the volunteer. Moreover, each of them had their own containers (no instructions were given apart from the bottle and mug/glass categories). Besides, the placement of the sensor around the glass, mug, or bottle was not fixed. Thus, this induces a high variance in the recorded data, as the reference system for the accelerometer and gyroscope signals can vary. For these reasons, we will provide a more detailed analysis of the proposal that articulates this work using this data set. The OHM data set is publicly available in [40].

A summary of the number of instances, classes, and the initial set of features that will be used as a reference for further comparison is shown in Table I.

B. Experimental Setup and Design

In this work, we have selected a variety of classic supervised ML classifiers that offer a good efficiency for resource-constrained environments when classifying sequences of continuous data from IoT sensors [9]. This selection includes logistic regression (LG), random forest (RF), KNN, Gaussian Naive Bayes (NB), Linear SVM, and MLP. It is important to remark that not all the included algorithms are probabilistic. In some cases, such as KNN or SVM, the confidence of the prediction is measured according to other parameters. These parameters correspond to the number of nearest neighbors from your new instance in KNN or the distance from

TABLE II
SELECTED EDGE DEVICES AND THEIR SPECIFICATIONS

Device	Processor	Architecture	CPU Freq	Cores	RAM
Laptop	i7-9750H	Intel x86	2.60 GHz	6	16 Gb
Nvidia Jetson Nano	Cortex-A57	ARMv8 64	1.43 GHz	4	2 Gb
Raspberry Pi 4 B	Cortex-A72	ARMv8 64	1.50 GHz	4	4 Gb
Raspberry Pi 3 B+	Cortex-A57	ARMv8 64	1.40 GHz	4	1 Gb
Raspberry Pi Zero	ARM11	ARMv6 32	1 GHz	1	512 Mb

the hyperplane in SVM. Moreover, RF is for itself an ensemble learning method, consisting of an ensemble of decision tree algorithms. This variety of learners enriches the scope of the proposal by not limiting it to specific probability-based algorithms and provides a broader scenario for evaluation purposes.

In this work, Edge devices are the target hardware platforms for inferring new knowledge through the different proposed strategies. Thus, we evaluate the performance of four different platforms and compare them against a laptop computer, which works as a reference device. The selected devices include one Nvidia Jetson Nano platform and three Raspberry Pi Foundation single-board minicomputers: 1) the Raspberry Pi model 4 B; 2) Raspberry Pi 3 model B+; and 3) the Raspberry Pi Zero W. Even though the three latter devices belong to the same family, there are significant differences in terms of processing power and memory, especially for the Raspberry Pi Zero, which is the most constrained device out of the five. Table II includes a summary of the main technical specifications of the selected platforms.

The experimental setup enabled for this comparative assessment entails a series of tasks that need to be performed to relate continuous signals to previously labeled actions. That is, the procedure of capturing/reading new data, processing it to obtain useful inputs, and comparing those inputs against already trained models. Thus, each of the instances of the data set goes through the following phases during the prediction process: 1) a 3-median data filtering stage which avoids anomaly values induced by noise; 2) a data segmentation process that divides each instance of data into five segments (or windows) of equal length without overlapping; 3) a feature calculation step that transforms raw data into statistical time-domain characteristics; 4) the normalization step that scales those features into an $[0-1]$ interval; and 5) the inference task itself. When evaluating the cascade strategy, steps 1)–3) are performed at the initial stage of the cascade. Then, the rest of the steps are repeated at each stage, but only those features that are needed for that level are calculated.

The classification solutions are powered by the ML framework Scikit-Learn [41]. All the evaluated devices are compatible with the same software libraries and are evaluated under the same conditions to provide a fair comparison between them. For the performance evaluation, we use the Python library Timeit to profile execution times. Furthermore, all experiments are executed solely on the CPU, with no other application running simultaneously. The materials used for the evaluation experiments are publicly available.[1]

[1]https://github.com/OihaneGomez/Model_Cascade_Optimization

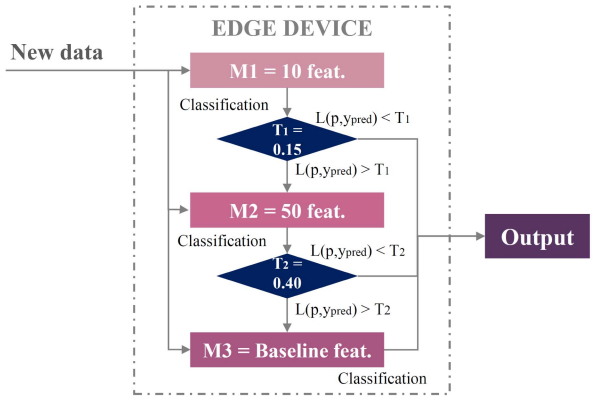

Fig. 5. Final configuration of the model cascade for assessing its potential to increase the efficiency of a stand-alone edge device.

C. Model Cascade Configuration

As the objective of this work is to evaluate the capabilities of the proposed approach in an algorithm and parameter independent and agnostic manner, we made a deliberate choice to maintain the same configuration for all the experiments in the comparison. Thus, no hyper-parameter tuning has been done to improve the ML algorithms' performance and adapt them to a specific scenario. This avoids the impact of other factors that could distort the overall vision of the strategy being evaluated.

In the proposed discriminative model cascade, the complexity of the different models is given by the number of features selected for creating the model, as it is a crucial factor for its computational cost [34]. For this reason, for every stage i a feature selection process is performed. This process reduces the feature matrix's dimension by removing irrelevant features, obtaining the subset of them that contributes the most to the prediction. From the exiting feature selection strategies, we apply a Chi2 filtering method [42], a computationally light strategy for ranking every characteristic of the signal according to its contribution to the prediction [43].

We opt for a cascade configuration based on three layers of independent models trained with an increasing number of features, as shown in Fig. 5. This cascade of models (that is, the three layers that it encompasses) is deployed on a single stand-alone Edge device to evaluate its performance when classifying new sequences of data. This performance is compared against a single reference model. For this reason, all the models of the cascade are trained using the same learning algorithm to provide a fair comparison between having a unique model and having a discriminative cascade of three models performing classification tasks on Edge devices. The first level, i_1, corresponds to a reduced model constituted by only ten features. This model initially classifies every new data sequence. This way, the most straightforward data sequences are processed in a computationally lighter way if their prediction meets the confidence threshold t_1. Recall that this confidence threshold represents the maximum log loss value for considering the output of the corresponding model as acceptable. In the second level, i_2, the number of features of the model increases up to 50. In this case, the model classifies only those sequences that the initial model was not able to classify with

the required confidence. Finally, the last level, i_3, is in charge of classifying all the remaining instances (those not classified at level i_1 and level i_2). This final model is trained using all the available baseline features. For comparison purposes, this configuration of layers and the number of features remain the same, independent from the data set or the evaluated learning algorithm.

The agnostic nature of this setup also applies to the thresholds t_i applied at each stage of the cascade. We have selected two different thresholds, one more restrictive than the other. Even though the log loss value is dependent on the number of classes and the balance of the data set, those thresholds remain the same in each of the evaluated data sets and algorithms. For the sake of comparison, the first threshold t_1 (from model 1 to 2) is set at 0.15, which is a highly restrictive value, as it is closer to 0. The second one t_2 (from model 2 to 3) is set at 0.40, being laxer than the previous one. These parameters represent the maximum log loss value for considering the output of the corresponding model as acceptable.

As aforementioned, we have fixed this selection of models and thresholds for all our experiments. Furthermore, we focus our evaluation on one specific configuration of the model cascade regarding its number of layers. Thus, the evaluated setup is only one example of the many alternative configurations that the proposed approach's flexibility allows. While tuning the models or defining the most suitable configuration of the cascade is out of the scope of this work, we hypothesize that both the selected number of features and the applied thresholds could provide a good cost-accuracy balance for our evaluation purposes. For this reason, we leave the possibility of better adjusting these factors or evaluating their impact open for further implementations of this strategy in which parameter fitting, thresholds adjustments, or selecting the most appropriate number of layers could be studied for the specific task and data. In the following, we will provide the results, both in terms of classification performance and efficiency, for the described setup.

V. Analysis and Results

Several experiments were conducted to analyze the potential benefits of the presented approach regarding the classification performance and efficiency of the proposal. For this reason, the results presented in this section can be divided into three main categories: 1) the evaluation of the classification accuracy for the presented approach; 2) the comparison of the three variations of the proposal described in Section III; and 3) the computational cost, in terms of execution time, of the selected variant when performing classification tasks in one piece of each of the evaluated platforms.

A. Classification Results

This section evaluates the accuracy of the model cascade according to the procedure described in Section IV. For evaluation purposes, we have adapted the training schemes explained in Section III for each variation of the cascade. To ensure the reliability of our experiments, a stratified five fold-cross-validation (CV) procedure was applied for each data set. The CV process is applied at each stage of the cascade independently.

In the parallel approach, the training set (composed of four folds of the data set) remains the same at every level. At each stage, only the samples of the test set (the remaining fold) whose log loss is above t_i go to the next stage. The evaluation metrics for the cascade are computed using the samples of the test folds. For each sample, the prediction obtained at the stage in which the sample got a log loss below t_i is taken into account to obtain the final classification score. That is, if the kth sample gets a log loss above t_1 at the first level and a log loss below t_2 at the second level, only the prediction of model 2 would be considered. For the sequential approach, the CV process of the first level is performed over the whole training set. For the next stage, at each iteration, only the samples of the test fold that got a log loss above t_1 are used. Then, a new fivefold CV is performed with all the data that remain unclassified from each test folds of the previous levels. This process is repeated with models 2 and 3. In the case of the hybrid approach, the parallel approach is used between the first and second models, while the sequential approach is used between the second and the third one. To increase the robustness of the evaluation, the results are provided as the average of 10 runs for each of the variations.

We calculate the macro-weighted F1-score metric for all the experiments, which provides the harmonic mean of precision and recall metrics. It averages both the true positives among all positive results (precision) and the correct labeled positives based on all the correct positive and negative events (recall).

We compare the three variations of the model cascade (sequential, parallel, and hybrid) against the reference model and the stacking technique. The reference model corresponds to a single model created using the maximum initial number of features available, according to Table I. Thus, this is the most complex single model that can be created using the available features. Stacking is an ensemble modeling technique in which a stack of different estimators is used to train a meta-predictor based on the outputs of all the individual estimators. Despite their differences, this is the closest example of model cascade among the rest of the model ensembling techniques. We use the same setup for the stacking method and the cascade, using 10, 50, and the total available features to train each level. In stacking, both the single models and the final meta-predictor are trained using the same algorithm.

Table III shows the obtained results for the five methodologies (reference model, stacking, and the three variations of the cascade) in terms of F1 macro results. Besides, Table IV presents the relative percentage of instances classified with the two initial models of the cascade (10 and 50 features). For illustrative purposes, these tables have been colored to better represent the magnitude of the obtained F1 results and classification percentages, being the reddish colors the lowest values and the bluish the highest ones. This provides an absolute representation of the performance of the different evaluated scenarios for all the data sets and algorithms combined.

A closer look at the algorithm-dependent results indicates that LG, RF, and SVM are the classifiers that show the most stable performance in all the evaluated data sets,

TABLE III
F1 MACRO RESULTS FOR THE REFERENCE MODEL, THE STACKING METHOD, AND THE THREE PRESENTED VARIATIONS OF THE MODEL CASCADE

DATASET	Threshold	Features	Method	Algorithm					
				LG	RF	KNN	NB	SVM	MLP
OHM	T1 = 0.15 T2 = 0.40	486 (all)	REFERENCE	**87.69 (SD 0.708)**	**89.44 (SD 0.368)**	80.03 (SD 0.596)	**78.76 (SD 0.122)**	87.35 (SD 0.542)	**87.75 (SD 0.407)**
			Stacking	87.58 (SD 0.900)	88.27 (SD 0.775)	72.45 (SD 1.817)	78.23 (SD 0.401)	**87.93 (SD 0.551)**	86.23 (SD 0.888)
		10-50-486	Casc. sequential	86.70 (SD 0.561)	88.36 (SD 0.865)	81.66 (SD 0.742)	69.74 (SD 0.905)	85.76 (SD 0.727)	84.38 (SD 0.851)
			Casc. parallel	85.94 (SD 0.599)	88.58 (SD 0.425)	**82.52 (SD 0.668)**	66.87 (SD 0.709)	86.40 (SD 0.682)	83.45 (SD 0.950)
			Casc. hybrid	86.93 (SD 0.806)	87.90 (SD 0.473)	81.93 (SD 1.334)	66.82 (SD 0.646)	85.96 (SD 0.692)	82.70 (SD 0.965)
ADL	T1 = 0.15 T2 = 0.40	162 (all)	REFERENCE	83.35 (SD 1.093)	85.92 (SD 0.948)	**86.45 (SD 1.166)**	**74.74 (SD 1.357)**	**88.96 (SD 0.951)**	**83.24 (SD 1.118)**
			Stacking	79.09 (SD 1.160)	83.66 (SD 1.111)	72.98 (SD 1.742)	53.89 (SD 3.183)	85.30 (SD 1.614)	77.73 (SD 1.451)
		10-50-162	Casc. sequential	84.39 (SD 0.922)	86.37 (SD 1.605)	79.24 (SD 1.001)	63.71 (SD 1.671)	86.98 (SD 1.075)	78.91 (SD 1.826)
			Casc. parallel	**84.48 (SD 1.000)**	85.78 (SD 0.663)	84.55 (SD 0.798)	65.41 (SD 1.709)	88.36 (SD 1.155)	80.27 (SD 1.790)
			Casc. hybrid	84.21 (SD 1.434)	**86.68 (SD 1.274)**	80.73 (SD 1.306)	65.41 (SD 1.709)	87.17 (SD 1.145)	77.56 (SD 2.299)
HAR	T1 = 0.15 T2 = 0.40	561 (all)	REFERENCE	97.96 (SD 0.065)	97.85 (SD 0.076)	**96.93 (SD 0.106)**	**73.90 (SD 0.357)**	**98.67 (SD 0.069)**	**98.19 (SD 0.137)**
			Stacking	**98.01 (SD 0.075)**	**98.01 (SD 0.075)**	95.67 (SD 0.245)	70.13 (SD 0.604)	98.66 (SD 0.066)	97.91 (SD 0.150)
		10-50-561	Casc. sequential	95.71 (SD 0.072)	95.77 (SD 0.138)	90.46 (SD 0.235)	73.87 (SD 1.131)	93.77 (SD 0.134)	92.73 (SD 0.342)
			Casc. parallel	95.45 (SD 0.079)	96.16 (SD 0.107)	92.12 (SD 0.199)	73.00 (SD 0.393)	94.06 (SD 0.101)	93.27 (SD 0.193)
			Casc. hybrid	95.50 (SD 0.077)	96.33 (SD 0.099)	91.49 (SD 0.203)	73.35 (SD 0.453)	93.67 (SD 0.072)	92.87 (SD 0.223)
OFFICE	T1 = 0.15 T2 = 0.40	272 (all)	REFERENCE	87.52 (SD 0.145)	**97.37 (SD 0.083)**	**94.93 (SD 0.107)**	63.86 (SD 0.059)	**93.37 (SD 0.169)**	**92.34 (SD 0.378)**
			Stacking	84.58 (SD 0.154)	94.23 (SD 0.296)	72.73 (SD 2.792)	**65.84 (SD 0.156)**	89.33 (SD 0.211)	84.15 (SD 0.684)
		10-50-272	Casc. sequential	**88.66 (SD 0.149)**	96.60 (SD 0.135)	92.94 (SD 0.188)	57.65 (SD 0.277)	92.48 (SD 0.186)	91.02 (SD 0.750)
			Casc. parallel	87.25 (SD 0.153)	96.81 (SD 0.102)	93.95 (SD 0.162)	54.65 (SD 0.213)	92.28 (SD 0.160)	92.19 (SD 0.284)
			Casc. hybrid	88.65 (SD 0.167)	96.47 (SD 0.108)	93.29 (SD 0.144)	54.85 (SD 0.213)	91.81 (SD 0.127)	91.56 (SD 0.397)
SPORTS	T1 = 0.15 T2 = 0.40	162 (all)	REFERENCE	**97.58 (SD 0.051)**	**99.13 (SD 0.037)**	**98.01 (SD 0.070)**	**94.06 (SD 0.108)**	**98.43 (SD 0.049)**	**98.05 (SD 0.122)**
			Stacking	97.47 (SD 0.062)	98.88 (SD 0.073)	93.85 (SD 1.779)	80.37 (SD 0.856)	98.08 (SD 0.050)	97.39 (SD 0.224)
		10-50-162	Casc. sequential	97.35 (SD 0.060)	98.57 (SD 0.089)	96.18 (SD 0.212)	77.10 (SD 1.531)	97.09 (SD 0.130)	96.16 (SD 0.177)
			Casc. parallel	97.30 (SD 0.058)	98.65 (SD 0.048)	96.64 (SD 0.192)	74.11 (SD 1.435)	97.26 (SD 0.105)	96.36 (SD 0.150)
			Casc. hybrid	97.40 (SD 0.041)	98.53 (SD 0.070)	96.68 (SD 0.210)	74.05 (SD 1.432)	97.14 (SD 0.137)	95.97 (SD 0.336)

TABLE IV
PERCENTAGE (%) OF INSTANCES CLASSIFIED WITH THE TWO INITIAL MODELS OF THE CASCADE FOR THE THREE PRESENTED VARIATIONS

DATASET	Threshold	Features	Method	Algorithm					
				LG	RF	KNN	NB	SVM	MLP
OHM	T1 = 0.15 T2 = 0.40	10-50-486	Casc. sequential	**70.42 (SD 1.981)**	**77.07 (SD 0.691)**	**80.31 (SD 0.917)**	**99.68 (SD 0.218)**	76.77 (SD 1.673)	**97.84 (SD 0.480)**
			Casc. parallel	70.21 (SD 0.670)	76.36 (SD 0.649)	79.98 (SD 1.006)	99.39 (SD 0.225)	**78.30 (SD 0.841)**	97.28 (SD 0.543)
			Casc. hybrid	70.21 (SD 0.670)	76.36 (SD 0.649)	79.98 (SD 1.006)	99.39 (SD 0.225)	**78.30 (SD 0.841)**	97.28 (SD 0.543)
ADL	T1 = 0.15 T2 = 0.40	10-50-162	Casc. sequential	**51.47 (SD 0.415)**	61.60 (SD 1.484)	74.80 (SD 1.098)	99.32 (SD 0.266)	61.07 (SD 1.337)	88.02 (SD 1.801)
			Casc. parallel	51.44 (SD 0.539)	**65.42 (SD 0.727)**	**79.30 (SD 0.734)**	**99.52 (SD 0.380)**	**62.69 (SD 0.862)**	**91.63 (SD 0.777)**
			Casc. hybrid	51.44 (SD 0.539)	**65.42 (SD 0.727)**	**79.30 (SD 0.734)**	**99.52 (SD 0.380)**	**62.69 (SD 0.862)**	**91.63 (SD 0.777)**
HAR	T1 = 0.15 T2 = 0.40	10-50-561	Casc. sequential	68.12 (SD 0.128)	74.76 (SD 0.199)	82.09 (SD 0.391)	**97.29 (SD 1.844)**	77.63 (SD 0.474)	84.23 (SD 3.179)
			Casc. parallel	**69.46 (SD 0.104)**	**81.67 (SD 0.171)**	**85.80 (SD 0.217)**	93.96 (SD 0.722)	**81.37 (SD 0.160)**	**86.37 (SD 0.544)**
			Casc. hybrid	**69.46 (SD 0.104)**	**81.67 (SD 0.171)**	**85.80 (SD 0.217)**	93.96 (SD 0.722)	**81.37 (SD 0.160)**	**86.37 (SD 0.544)**
OFFICE	T1 = 0.15 T2 = 0.40	10-50-272	Casc. sequential	**65.40 (SD 0.065)**	87.18 (SD 0.204)	92.64 (SD 0.307)	**99.73 (SD 0.032)**	74.13 (SD 0.214)	90.59 (SD 3.742)
			Casc. parallel	65.27 (SD 0.156)	**87.41 (SD 0.244)**	**93.66 (SD 0.210)**	99.41 (SD 0.058)	**76.02 (SD 0.313)**	**92.72 (SD 0.497)**
			Casc. hybrid	65.27 (SD 0.156)	**87.41 (SD 0.244)**	**93.66 (SD 0.210)**	99.41 (SD 0.058)	**76.02 (SD 0.313)**	**92.72 (SD 0.497)**
SPORTS	T1 = 0.15 T2 = 0.40	50-50-162	Casc. sequential	**81.17 (SD 1.071)**	94.67 (SD 0.123)	96.51 (SD 0.165)	**99.76 (SD 0.144)**	96.38 (SD 0.127)	**97.58 (SD 0.172)**
			Casc. parallel	74.99 (SD 0.192)	**94.79 (SD 0.075)**	**97.79 (SD 0.132)**	99.41 (SD 0.135)	**96.64 (SD 0.084)**	97.23 (SD 0.261)
			Casc. hybrid	74.99 (SD 0.192)	**94.79 (SD 0.075)**	**97.79 (SD 0.132)**	99.41 (SD 0.135)	**96.64 (SD 0.084)**	97.23 (SD 0.261)

presenting uniform F1 results. Ensemble techniques obtained worse results in MLP and KNN compared to the reference model. They also show high variances compared to the rest of the algorithms, particularly for the ADL and HAR data sets. Still, they maintain reasonable classification rates. NB obtains the worst performance. Not only the reference results for NB are the worst but applying NB within the cascade penalizes its results. A reason for this circumstance can be found in Table IV. Here, NB obtains, by far, the highest rates of classified instances using the simplest models (first levels of the cascade), in which almost all the instances do not reach the last level. This has an impact on the classification results and may indicate that the applied confidence thresholds are not well calibrated for this algorithm. Similarly, MLP and, to a lesser degree, KNN also classify a high number of instances at the early stages of the cascade. As it can be observed, for the last two algorithms, there is an indirect effect between the correlation of classified instances and the F1 values, meaning that the higher this percentage is, the lower the classification result may be.

Tables V and VII show a summarized comparison between the different methodologies (the reference model, the stacking technique, and the three variations of the cascade). The former averages the algorithms' results, while the latter averages the results across all the algorithms and data sets. They also include the averaged percentage decrease of comparing the stacking and the three cascade approaches against the reference model. The percentage decrease measures the percent change between two values, according to

$$\%D = \frac{\text{Initial Value} - \text{Final Value}}{\text{Initial Value}} * 100. \tag{2}$$

In this case, $\%D$ measures the relative decrease of each of proposed approach's results (FinalValue) when compared against the reference model (InitialValue).

TABLE V
AVERAGE F1 RESULTS FOR ALL THE CLASSIFIERS AND THEIR DECREASE PERCENTAGE (% D) COMPARED AGAINST THE REFERENCE RESULTS

	OHM		ADL		HAR		OFFICE		SPORTS	
	F1	% D	F1	% D	F1	% D	F1	% D	F1	% D
REFERENCE	**86.45**		**85.58**		**97.92**		**93.11**		**98.24**	
Stacking	83.45	2.02	75.44	9.95	93.07	0.91	81.81	7.28	94.34	3.28
Sequential	82.77	2.82	79.93	4.59	90.39	3.76	86.56	1.90	93.74	3.90
Parallel	82.29	3.38	81.48	2.75	90.68	3.45	86.19	2.32	93.39	4.26
Hybrid	82.04	3.68	80.29	4.16	90.54	3.60	86.11	2.41	93.30	4.36

TABLE VI
AVERAGE PERCENTAGE (%) OF INSTANCES CLASSIFIED WITH THE TWO SIMPLEST MODELS

	OHM	ADL	HAR	OFFICE	SPORTS
Sequ.	**83.68 (SD 12.1)**	72.71 (SD 18.2)	80.69 (SD 9.9)	84.95 (SD 12.7)	94.34 (SD 6.7)
Parallel	83.59 (SD 11.9)	**75.00 (SD 18.4)**	**83.10 (SD 8.1)**	**85.75 (SD 12.8)**	**93.47 (SD 9.2)**
Hybrid	83.59 (SD 11.9)	**75.00 (SD 18.4)**	**83.10 (SD 8.1)**	**85.75 (SD 12.8)**	**93.47 (SD 9.2)**

TABLE VII
AVERAGE F1 RESULTS AND THEIR DECREASE PERCENTAGE (% D) FOR ALL THE LEARNERS AND DATA SETS

	F1	% D
REFERENCE	**89.73 (SD 5.85)**	
Stacking	85.62 (SD 7.97)	4.69 (SD 3.80)
Sequential	86.68 (SD 5.58)	3.39 (SD 1.04)
Parallel	86.80 (SD 5.18)	3.23 (SD 0.74)
Hybrid	86.45 (SD 5.50)	3.64 (SD 0.75)

In this regard, the single (and more complex) model that acts as the reference method outperforms the rest of the candidates (stacking and cascade strategies). That is, stacking and cascade strategies obtain, on average, lower results than the reference model. Moreover, cascade variations show similar or even higher classification results than stacking techniques across algorithms. In fact, when compared against the reference model, the average percentage decrease when averaging the 3 cascade variations is 3.42% (SD 0.81%), compared to the 4.69% (SD 3.8%) shown by stacking techniques. Nonetheless, based on the standard deviation, stacking techniques present higher performance variations depending on the data set. On the contrary, cascade methods present more consistent results.

However, it is paramount to emphasize that the importance of cascade approaches lies in the optimization possibilities they enable and their potential efficiency, not in their capabilities to improve the classification results. Therefore, its interest lies in the relative comparison of how much computational improvement can be obtained without substantially impairing the system's performance. With the cascade approach, the potential computational gains are given by the percentage of instances classified with the simplest models, summarized in Table VI. The first noticeable fact is that parallel and hybrid approaches classify the same number of instances with the two first stages of the model cascade. This is attributed to both of them sharing the same structure for those levels. Beyond that, some differences can be observed depending on the data set, but, on average, the 83.88% (SD 13.17%) of the instances can be classified within these two stages.

When combined, the average F1-score (Table V) and the average percentage of classified instances with the initial levels of the cascade (Table VI) illustrate the capabilities of the model cascade system and the benefits of setting a confidence threshold. This is reflected in the comparison of cascading variations with stacking techniques in Table VII. In the cascade method, setting a confidence threshold compensates for the potential errors induced by the lighter models. Thus, even mainly using simple models belonging to the first levels of the cascade, it is possible to achieve similar or even better results than using a much heavier and complex model such as the one created in the stacking technique. In our case, the studied cascade strategies are able to reduce by more than 80% the number of instances to be classified with the last level of the cascade and maintain detection rates that only decrease on approximately 3.4%. This means that assuming a decrease in accuracy lower than 4%, we can have a system in which more than 80% of the instances can be classified with much simpler models than the reference one.

B. Selecting the Best Strategy

Selecting which of the three variations of the cascade approach is better requires choosing between maximizing the classification results or the number of instances classified with the simplest models (recall that efficiency in this piece of work is given by the number of instances that are classified with the initial levels of the cascade). Depending on this choice, the performance of the system is prioritized in terms of the classification outcomes or its efficiency, one being conditional on the other. Having two different optimization objectives, the best strategy consists of analyzing which of the three variations provides a better balance between both factors (i.e., which of the three variations optimize in a better balance between them).

By comparing Tables V and VI some initial conclusions can be obtained in this regard. At least for the evaluated data sets, the hybrid approach does not provide any advantage over the rest of the methods. Not only the parallel and sequential approaches obtain, in general, better classification results but the models with the parallel approach classify the same percentage of instances as models with the hybrid approach, as they maintain the same structure for the cascade's initial stages. From this point, further evaluation can optimize this tradeoff to find which is the model cascade variation that better fulfills those two objectives (the best classification results and highest number of initially classified instances). To that end, we conduct a Pareto multiobjective optimization analysis [44]. The Pareto analysis is a statistical tool for decision making that selects a limited number of Pareto-optimal solutions, corresponding to the Pareto front, which maximizes two objectives: 1) accuracy and 2) the percentage of classified instances in the first stage of the cascade in our case.

First, we apply this strategy for all the combinations of algorithms and data sets we are considering in this work. Additionally, we have specifically targeted the OHM Data set as we will focus the rest of the computational experiments on it. Fig. 6 represents the relationship between the accuracy (F1 result) and the percentage of classified instances, both for the OHM Data set (left plot) and the combination of all the data sets (middle and right plots). A zoom-in of the upper right corner of the middle plot is shown in the right plot for the sake of visualization. This region represents the best balance between the two analyzed factors. Those plots also include the Pareto frontier as a dashed line representing the optimal solutions that maximize the accuracy (highest y-axis values) and the classification percentage (largest values on the x-axis). Finally, Table VIII shows each of these optimal solutions for all the data sets and the OHM data set. It includes the name of their corresponding variation of the cascade

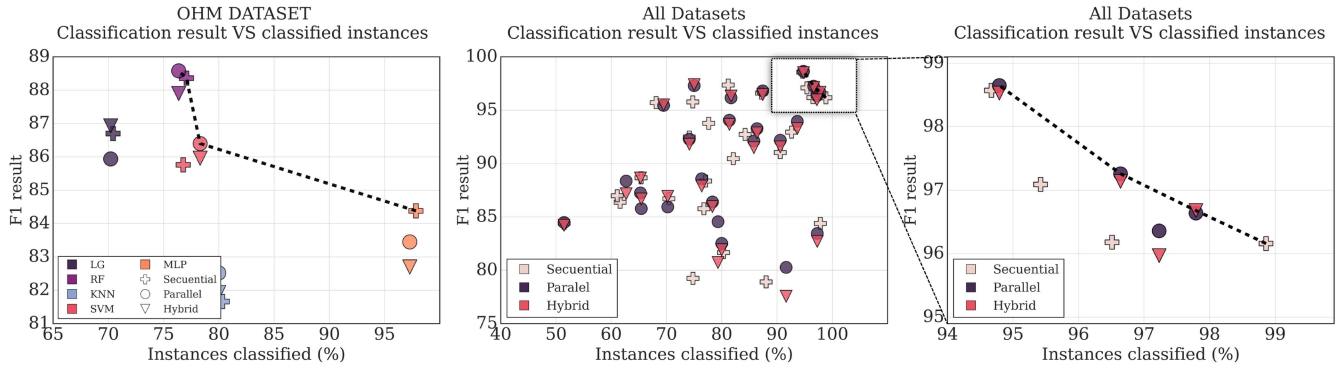

Fig. 6. Balance between the classification results and the number of instances classified in the first levels of the cascade for the OHM data set (left plot) and all the data sets together (middle plot) with a special focus on the best accuracy-classified percentage region of the latter (right plots).

TABLE VIII
FINAL PARETO OPTIMAL SOLUTIONS

All Datasets			OHM Dataset		
Type	F1	%	Type	F1	%
Parallel	98.65	94.79	Parallel	88.58	76.36
Parallel	97.26	96.64	Sequential	88.36	77.07
Hybrid	96.68	97.79	Parallel	86.40	78.30
Sequential	96.16	98.86	Sequential	84.38	97.84

approach, F1 value, and the percentage of initially classified instances.

The Pareto optimal solutions listed in Table VIII show that the sequential and parallel approaches are predominant, being the latter the most frequently listed variation. Although it is not possible to generalize based on the evaluation carried out and the five selected data sets, it does allow us to obtain certain indications of the potential strengths of this approach. For this reason, we carry out the computational experiments using this variation, as it obtains the most balanced results. Either way, considering how close the average classification percentage is for the sequential and parallel approaches (83.68% versus 83.59%) in Table VI, no substantial performance differences are expected due to this decision.

C. Timing Results

The results presented in this section seek to shed light on the potential of the cascade strategy to save time and resources in the classification task. That is, to make a more efficient usage of the available resources of each individual node deployed at the Edge. To evaluate its impact, we compare the performance of both the reference model and the parallel implementation of the model cascade strategy, being this latter one selected after performing the Pareto optimization process, using the OHM data set. We measure the computational cost as the actual amount of time needed to process and classify new instances of data according to the experimental setup described in Section IV. In particular, we compare the performance of the cascade strategy against a single reference model and evaluate the potential computational savings (in terms of time) of each of these approaches when deployed on a single Edge device (i.e., one piece of the evaluated platforms).

We have randomly divided the data set into two parts to perform this experiment, maintaining the 80%–20% proportion used in the fivefold CV process. Thus, 800 out of the 1000 instances that the data set contains are used for training and the remaining group of data, consisting of 200 instances, are used for prediction. Therefore, we compare the elapsed time in classifying these 200 instances of the data set with the reference model and the cascade strategy. This test resembles an online classification scenario in which new sequences of continuous data need to be predicted by the intelligent system. To ensure the reliability of the results, the averaged elapsed time is computed (i.e., the experiment is executed 30 times—10 runs of 3 loops).

The results obtained in the experimental evaluation for each of the five selected devices operating individually are included in Table IX. It shows that, in the evaluated scenario, our approach is more efficient than the reference model when classifying new instances of HAR data. This also means that classifying part of these instances with simpler models offsets the possible overhead of some of them going through the different stages of the cascade until the confidence requirement is met. Table X summarizes the order of these time improvements, reflected in the decrease percentage (%D) between the reference times and the optimized ones. Again, this table has been colored to better represent the magnitude of obtained time improvements, being the reddish colors the lowest values and the bluish the highest ones. As can be observed, in most of the cases, the metrics improve more than the 50%. As expected, the obtained results for every algorithm are highly reliant on the percentage of classified instances of the OHM data set showed in Table IV. Consequently, those algorithms that could deal with a larger amount of data at the early stages of the model cascade show higher performance improvements. In the case of RF, the time complexity of the prediction process associated with this algorithm penalizes the efficiency of the cascade approach, obtaining improvements below the ones that would correspond to its classification percentage. Even so, it still improves the reference model's performance.

When averaging all the algorithms, the time improvements for the first two devices were 61.67% (SD 15.71%) for the laptop and 64.93% (SD 18.01%) for the Jetson Nano board. For the Raspberry foundation devices, the average Pi 4 results were 60.56% (SD 17.23%), the Pi 3 obtained 60% (SD

TABLE IX
ELAPSED TIME FOR CLASSIFYING 200 INSTANCES OF NEW DATA WITH THE REFERENCE MODEL AND THE PARALLEL CASCADE METHOD

		OHM Dataset					
		LG	**RF**	**KNN**	**NB**	**SVM**	**MLP**
Laptop	Reference	$7.69s \pm 156ms$	$9.46s \pm 181ms$	$7.96s \pm 198ms$	$7.29s \pm 122ms$	$7.79s \pm 54.8ms$	$7.19s \pm 331ms$
	Cascade	$4.1s \pm 198ms$	$5.38s \pm 57.4ms$	$3.08s \pm 193ms$	$1.21s \pm 33.3ms$	$3.12s \pm 34.3ms$	$1.76s \pm 59.6ms$
Jetson Nano	Reference	$39s \pm 242ms$	$46.5s \pm 131ms$	$39.6s \pm 14.5ms$	$38.7s \pm 7.34ms$	$38.8s \pm 1.34ms$	$38.8s \pm 15.8ms$
	Cascade	$18.9s \pm 20.9ms$	$27.3s \pm 5.27ms$	$14.6s \pm 3.83ms$	$6.12s \pm 2.35ms$	$14.7s \pm 6.57ms$	$4.91s \pm 1.33ms$
RPi 4B	Reference	$35.7s \pm 14.4ms$	$42.4s \pm 63.4ms$	$36.8s \pm 7.44ms$	$36s \pm 3.72ms$	$36s \pm 4.93ms$	$35.8s \pm 5.14ms$
	Cascade	$17.1s \pm 3.27ms$	$23.9s \pm 31.2$	$13.3s \pm 3.63ms$	$5.37s \pm 142s$	$13s \pm 6.26ms$	$8.6s \pm 2.88ms$
RPi 3B+	Reference	$1min10s \pm 301ms$	$1min17s \pm 121ms$	$1min11s \pm 152ms$	$1min10s \pm 164ms$	$1min11s \pm 233ms$	$1min11s \pm 233ms$
	Cascade	$39s \pm 86.6ms$	$44.3s \pm 161ms$	$29.6s \pm 243ms$	$14.5s \pm 35.9ms$	$31.4s \pm 280ms$	$11.9s \pm 11.8ms$
RPI Zero	Reference	$6min55s \pm 584ms$	$7min46s \pm 265ms$	$7min1s \pm 73.5ms$	$6min57s \pm 100ms$	$6min58s \pm 49.6ms$	$6min56s \pm 524ms$
	Cascade	$4min13s \pm 76ms$	$4min14s \pm 156ms$	$2min44s \pm 38.8ms$	$1min16s \pm 186ms$	$2min47s \pm 185ms$	$2min17s \pm 79.6ms$

TABLE X
DECREASE PERCENTAGE (% D) SHOWING THE TIME IMPROVEMENTS WHEN APPLYING THE CASCADE METHOD

	OHM Dataset					
	LG	**RF**	**KNN**	**NB**	**SVM**	**MLP**
Laptop	46.68	43.13	61.31	83.40	59.95	75.52
Jetson Nano	51.54	41.29	63.13	84.19	62.11	87.35
Rpi 4B	52.10	43.36	63.86	85.08	62.22	75.98
RPi 3B+	44.29	42.47	58.31	79.29	55.77	83.24
Rpi Zero	48.19	45.49	61.05	81.77	60.05	67.07

13.21%), and the Pi Zero 62,31% (SD 13.21%). Therefore, the time improvement explored in this section produced an enhanced performance in all the classification tasks regardless of the devices' resources for a continuous HAR classification system. Furthermore, the time decreased on average 62.31% (SD 1.96%) when averaging all the devices' mean results. These results represent the actual amount of time that the evaluated operations require to be completed as well as its reduction when applying the proposed cascade strategy. According to Bennett and Parrado-Hernández [45], those times are related to obtaining good performance in practice in terms of execution times. On the one hand, this increases response times in IoT systems, as data is processed faster. On the other hand, it minimizes the complexity of intelligent systems and increases its scalability, allowing to deploy more complex HAR applications in a local stage that, otherwise, could not be efficiently executed at the Edge.

D. Limitations

Before concluding this section, we want to address the possible limitations of this study. The obtained results are dependent on the number of levels and the selection of the models (according to the number of features in each stage) that constitute the cascade. Besides, they are also dependent on the probabilistic thresholds that govern the transition from one stage to another. In these experiments, the same parameters are applied for every algorithm and data set to provide better insights into the effect of the different strategies in the classification rates. Additionally, they do not cover other cascade settings in terms of model parameters or number layers. For this reason, this evaluation is not intended to test every possible combination of models and attributes but to provide insights on the optimization possibilities of this strategy based on the described evaluation procedure. Therefore, the importance of the performed comparison lies in the relative observable results between the different strategies, not in their absolute values. It is important to note that the results obtained, both in the reference model and in the strategies evaluated, are not the best possible ones, nor do they seek to be compared against the state-of-the-art of their respective data sets.

Besides, despite the possibilities that separating a complex model into a simpler model enables, this evaluation does not cover a distributed scenario in which cascade layers are deployed in various devices. Thus, the provided timing results and performance may vary if the system's setup is changed. Finally, the obtained computational times are not deterministic and can be sensitive to the device's internal processes or routines (despite calculating these times iteratively and running solely on the device). To increase the reliability of the measurements, we took a large number of measurement samples and provided average results.

VI. DISCUSSION

The uptake of emerging technologies and the unleashing of innovation when monitoring and predicting the environment are the drivers of an interconnected future that can improve many aspects of our lives in domains, such as the city [46], the workplace [47], or our home [48]. Moreover, AI techniques are transforming our world and have the potential to enhance health [49], promote sustainable practices [50], or lead us toward achieving the Sustainable Development Goals [51]. These technologies have brought with them significant advances in innovation, creating profound impacts on society. As an example, identifying and analyzing the occupants' activities and behaviors toward energy utilization in buildings has an energy-saving potential of up to 30%, which has a direct positive effect in reducing climate change risks [52].

However, the environmental impact caused by the infrastructure necessary to operate these kinds of solutions is rarely taken into account. For this reason, there is a need for understanding their benefits while mitigating their risks. On the one hand, users demand more and better services without really being aware of the energy implications. From a user

perspective, it is easy to associate everyday activities such as turning on a light with the fact that it generates energy consumption. Nevertheless, in the digital world, this barrier is more blurred when it comes to the processes behind the services we use on a daily basis. This hinders understanding the magnitude of the resources needed for deploying some of the applications or services we use every day. On the other hand, the scientific world continues its progress toward the development of new techniques that improve existing ones and provide answers to more and more complex problems. Although this works in favor of progress, it does not take into consideration the high computational requirements of such advances.

Enabling technologies, such as Big Data or Deep Learning are based on increasingly complex mathematical models with millions of parameters that provide answers to complicated problems of image, voice, continuous signals, or text classification. The rise in popularity of Deep Learning, a subfield of AI that depends on processing vast amounts of data, has accentuated this sustainable problem in recent years due to its growing use and deployment [53]. Also, the Blockchain technology and cryptocurrencies have raised the debate about the viability of this exponential growth and the energy consumption and carbon emissions they bring with them. In fact, their massive energy consumption has reached a point in which it has become a primary environmental concern [54]. Bitcoin, the most famous Blockchain-based cryptocurrency, has an estimated electricity consumption compared with countries, such as Ireland, Hong Kong, or Austria [55]. This is a clear example of how a technological advance with enough potential to revolutionize the digital future can end up having more environmental costs than benefits.

Nonetheless, users and several sectors will continue demanding and adopting new services. In parallel, scientists need to keep pushing forward new horizons toward the convergence of different technologies to improve those services. Edge Computing and, in particular, embedded ML techniques propose a total paradigm shift to overcome this situation. For years, the world of AI and Deep Learning has been based on the search for massive amounts of data for training bigger models, scaling as much as possible the resources needed to deal with the complexity of those approaches. This has led to a centralized approach in which all captured data must be sent to remote data centers where the only equipment capable of deploying these systems is hosted. Those data centers are located hundreds or even thousands of kilometers away, which increases the associated costs of data transmission.

In contrast, in optimization approaches, the importance lies in the quality of the data and the efficiency of the models under the principle that any unnecessary data sent and/or processed is a potential waste of resources. As the complexity of classification techniques decreases, less powerful equipment is needed, and a local resource management is possible. For this reason, with this work, we want to shed light on the importance of making an optimal management of resources, avoiding as much as possible the high energy cost of classification tasks and the dependence on long-distance external servers. With the proposed cascade of models, classification times are reduced, which can be translated directly into lower energy use associated with such data processing and decreased latency times. Furthermore, splitting the reference model into straightforward, simpler, and independent models enables a more efficient usa of the available resources, alleviating the overhead of Edge nodes. This way, it prevents energy-intensive and complex models from taking on tasks for which they are oversized. At the same time, this helps to scale up the complexity of the applications that can be allocated in those resource-constrained devices. This way, splitting the computation opens the door to a greater flexibility in the architectures surrounding the IoT devices. It also allows deploying those techniques in low-powered Edge devices (e.g., microcontrollers), which reduces the dependence on large infrastructures and remote data centers. Thanks to the ever-increasing computational capacity of hardware devices, this strategy could eventually enable a local system in which a preliminary classification step could be performed on end-point devices. At the same time, only some specific data would be sent to more powerful devices at the Edge.

We believe that further exploring the synergy between Edge Computing, IoT, and energy-efficient AI techniques may be a suitable alternative toward a more sustainable digital world where technological advances do not imply compromising future environmental development.

VII. Conclusion and Future Work

In this work, we have presented a method to optimize and improve the performance of classification tasks using model ensemble techniques in resource-constrained devices. This is motivated by the challenge of contributing to a more sustainable IoT development by understanding the need for simplified processing techniques that improves the usability and efficiency of Edge Intelligence solutions for different application scenarios with embedded AI. To that end, we have analyzed the suitability of three variations of a discriminative model cascade strategy that combine different classification stages to better fit the system to the complexity of the input data. Besides, we have empirically analyzed and compared the performance of the presented approach concerning HAR applications through four resource-constrained Edge platforms.

On average, the proposed model cascade strategy outperforms the results of the model stacking technique. Compared with the full (and more complex) reference model, it maintains acceptable detection rates with an average decline of only 3.42% while classifying around 80% of the instances at the early stages of the cascade. In the evaluated case of study, the proposed cascade strategy reduced by more than 60% the baseline processing times of the prediction task for a set of 200 instances in all the evaluated platforms. The outcome of this evaluation shows the potential of the proposed approach to substantially improve the performance of classification tasks without drastically compromising their results. It paves the way for low-latency and energy-efficient applications that make the Edge truly intelligent. This analysis seeks to assist in providing a strategy for optimizing similar problems

and contribute to bringing advanced processing capabilities to the Edge of the network.

Having evaluated the suitability of this approach to increase the efficiency of stand-alone Edge devices, the obtained results provide us with an outlook for the future work and the possibilities of adapting this scheme to a collaborative approach. The division of models that the cascade proposes enables new offloading schemes in the distribution of computing power around nearby devices around the Edge. Further research in this direction would be needed, analyzing the suitability of a hierarchy of heterogeneous Edge nodes to accommodate each of the cascade levels and identifying the most optimal communication mechanisms to promote this collaboration efficiently. In this regard, the future work also points to the possibility to port models with lower computational requirements from the Edge to end-device and allocate part of the cascade system there. Finally, studying the deployment of the proposed solution in a real-world online classification scenario would be interesting to extend the conclusions regarding its potential to meet the Edge requirements in a timely manner.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support of the Basque Governments Department of Education for the predoctoral funding of one of the authors and the Deustek Research Group. They also acknowledge NVIDIA Corporation for the donation of the Jetson Nano used for this research.

REFERENCES

[1] J. Wang, M. K. Lim, C. Wang, and M.-L. Tseng, "The evolution of the Internet of Things (IoT) over the past 20 years," *Comput. Ind. Eng.*, vol. 155, May 2021, Art. no. 107174.

[2] M. Asif-Ur-Rahman *et al.*, "Toward a heterogeneous mist, fog, and cloud-based framework for the Internet of Healthcare Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4049–4062, Jun. 2019.

[3] L. Hou *et al.*, "Internet of Things cloud: Architecture and implementation," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 32–39, Dec. 2016.

[4] B. Rana, Y. Singh, and P. K. Singh, "A systematic survey on Internet of Things: Energy efficiency and interoperability perspective," *Trans. Emerg. Telecommun. Technol.*, to be published.

[5] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[6] A. J. Ferrer, J. M. Marquès, and J. Jorba, "Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing," *ACM Comput. Surveys*, vol. 51, no. 6, pp. 1–36, 2019.

[7] S. Yu, X. Chen, L. Yang, D. Wu, M. Bennis, and J. Zhang, "Intelligent edge: Leveraging deep imitation learning for mobile edge computation offloading," *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 92–99, Feb. 2020.

[8] S. Yu, X. Chen, Z. Zhou, X. Gong, and D. Wu, "When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5G ultra-dense network," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2238–2251, Feb. 2021.

[9] S. Dhar, J. Guo, J. Liu, S. Tripathi, U. Kurup, and M. Shah, "On-device machine learning: An algorithms and learning theory perspective," 2019. [Online]. Available: arXiv:1911.00623.

[10] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[11] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7457–7469, Aug. 2020.

[12] O. Gómez-Carmona, D. Casado-Mansilla, F. A. Kraemer, D. L. Ipiña, and J. García-Zubia, "Exploring the computational cost of machine learning at the edge for human-centric Internet of Things," *Future Gener. Comput. Syst.*, vol. 112, pp. 670–683, Nov. 2020.

[13] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.

[14] R. E. Schapire, "The strength of weak learnability," *Mach. Learn.*, vol. 5, no. 2, pp. 197–227, 1990.

[15] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, 1992.

[16] C. Zhang and Y. Ma, *Ensemble Machine Learning: Methods and Applications*. New York, NY, USA: Springer, 2012.

[17] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, 2001, p. 1.

[18] S. Xu, Q. Tang, L. Jin, and Z. Pan, "A cascade ensemble learning model for human activity recognition with smartphones," *Sensors*, vol. 19, no. 10, p. 2307, 2019.

[19] J. Wu, J. M. Rehg, and M. D. Mullin, "Learning a rare event detection cascade by direct feature selection," in *Proc. 16th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Whistler, BC, Canada, 2003, pp. 1523–1530.

[20] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Interdiscipl. Rev. Data Min. Knowl. Disc.*, vol. 8, no. 4, 2018, Art. no. e1249.

[21] M. Rokicki and M. Drozda, "Evaluating trade-offs in energy-efficient error detection," *Int. J. Commun. Syst.*, vol. 30, no. 7, 2017, Art. no. e3028.

[22] C. Kaynak and E. Alpaydin, "Multistage cascading of multiple classifiers: One man's noise is another man's data," in *Proc. ICML*, 2000, pp. 455–462.

[23] E. J. Silva, A. S. Britto, L. S. Oliveira, F. Enembreck, R. Sabourin, and A. L. Koerich, "A two-step cascade classification method," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, 2017, pp. 573–580.

[24] Z. Xu, M. J. Kusner, K. Q. Weinberger, M. Chen, and O. Chapelle, "Classifier cascades and trees for minimizing feature evaluation cost," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 2113–2144, 2014.

[25] M. Chen, Z. Xu, K. Weinberger, O. Chapelle, and D. Kedem, "Classifier cascade for minimizing feature evaluation cost," in *Proc. Artif. Intell. Stat.*, 2012, pp. 218–226.

[26] G. Fumera, F. Roli, and G. Giacinto, "Reject option with multiple thresholds," *Pattern Recognit.*, vol. 33, no. 12, pp. 2099–2101, 2000.

[27] L. S. Oliveira, A. Britto, and R. Sabourin, "Improving cascading classifiers with particle swarm optimization," in *Proc. IEEE 8th Int. Conf. Doc. Anal. Recognit. (ICDAR)*, 2005, pp. 570–574.

[28] P. Zhang, T. D. Bui, and C. Y. Suen, "A novel cascade ensemble classifier system with a high recognition performance on handwritten digits," *Pattern Recognit.*, vol. 40, no. 12, pp. 3415–3429, 2007.

[29] X. Wang, D. Kondratyuk, K. M. Kitani, Y. Movshovitz-Attias, and E. Eban, "Multiple networks are more efficient than one: Fast and accurate models via ensembles and cascades," 2020. [Online]. Available: arXiv:2012.01988.

[30] E. Park *et al.*, "Big/little deep neural network for ultra low power inference," in *Proc. IEEE Int. Conf. Hardw. Softw. Codesign Syst. Synth. (CODES+ISSS)*, 2015, pp. 124–132.

[31] S. Leroux *et al.*, "The cascading neural network: Building the Internet of Smart Things," *Knowl. Inf. Syst.*, vol. 52, no. 3, pp. 791–814, 2017.

[32] B. Taylor, V. S. Marco, W. Wolff, Y. Elkhatib, and Z. Wang, "Adaptive deep learning model selection on embedded systems," *ACM SIGPLAN Notices*, vol. 53, no. 6, pp. 31–43, 2018.

[33] L. Lefakis and F. Fleuret, "Joint cascade optimization using a product of boosted classifiers," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 1315–1323.

[34] O. Gómez-Carmona, D. Casado-Mansilla, D. L. Ipiña, and J. García-Zubia, "Simplicity is best: Addressing the computational cost of machine learning classifiers in constrained edge devices," in *Proc. 9th Int. Conf. Internet Things*, 2019, pp. 1–8.

[35] B. Bruno, F. Mastrogiovanni, and A. Sgorbissa, "A public domain dataset for ADL recognition using wrist-placed accelerometers," in *Proc. 23rd IEEE Int. Symp. Robot Human Interact. Commun.*, 2014, pp. 738–743.

[36] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," in *Proc. ESANN*, vol. 3, 2013, p. 3.

[37] J. A. A. Gonzalez. (2019). *Data for: Office Activity Recognition Using Accelerometers and Gyroscopes Located on the Forearms*. [Online]. Available: https://data.mendeley.com/datasets/2ws2g3pn6w

[38] K. Altun, B. Barshan, and O. Tunçel, "Comparative study on classifying human activities with miniature inertial and magnetic sensors," *Pattern Recognit.*, vol. 43, no. 10, pp. 3605–3620, 2010.

[39] C. R. Banbury *et al.*, "Benchmarking TinyML systems: Challenges and direction," 2020. [Online]. Available: arXiv:2003.04821.
[40] O. Gómez-Carmona and D. Casado-Mansilla. *Office Hydration Monitoring (OHM) Dataset*. Accessed: Apr. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4681206
[41] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html
[42] H. Liu and R. Setiono, "CHI2: Feature selection and discretization of numeric attributes," in *Proc. 7th IEEE Int. Conf. Tools Artif. Intell.*, 1995, pp. 388–391.
[43] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Elect. Eng.*, vol. 40, no. 1, pp. 16–28, 2014.
[44] C. Qian, Y. Yu, and Z.-H. Zhou, "Subset selection by pareto optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1774–1782.
[45] K. P. Bennett and E. Parrado-Hernández, "The interplay of optimization and machine learning research," *J. Mach. Learn. Res.*, vol. 7, pp. 1265–1281, Jan. 2006.
[46] R. Sánchez-Corcuera *et al.*, "Smart cities survey: Technologies, application domains and challenges for the cities of the future," *Int. J. Distrib. Sensor Netw.*, vol. 15, no. 6, pp. 1–19, 2019.
[47] O. Gómez-Carmona, D. Casado-Mansilla, and J. García-Zubia, "Opportunities and challenges of technology-based interventions to increase health-awareness in the workplace," *Transf. Ergon. Pers. Health Intell. Workplaces*, vol. 25, p. 33, Sep. 2019.
[48] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of Internet of Things for smart home: Challenges and solutions," *J. Cleaner Prod.*, vol. 140, pp. 1454–1464, Jan. 2017.
[49] H. H. Nguyen, F. Mirza, M. A. Naeem, and M. Nguyen, "A review on IoT healthcare monitoring applications and a vision for transforming sensor data into real-time clinical feedback," in *Proc. IEEE 21st Int. Conf. Comput. Supported Cooperat. Work Design (CSCWD)*, 2017, pp. 257–262.
[50] A. Irizar-Arrieta, O. Gómez-Carmona, A. Bilbao-Jayo, D. Casado-Mansilla, D. López-De-Ipiña, and A. Almeida, "Addressing behavioural technologies through the human factor: A review," *IEEE Access*, vol. 8, pp. 52306–52322, 2020.
[51] M. A. Goralski and T. K. Tan, "Artificial intelligence and sustainable development," *Int. J. Manag. Educ.*, vol. 18, no. 1, 2020, Art. no. 100330.
[52] Y. Zhang, X. Bai, F. P. Mills, and J. C. Pezzey, "Rethinking the role of occupant behavior in building energy performance: A review," *Energy Build.*, vol. 172, pp. 279–294, Aug. 2018.
[53] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," 2019. [Online]. Available: arXiv:1906.02243.
[54] J. Truby, "Decarbonizing bitcoin: Law and policy choices for reducing the energy consumption of blockchain technologies and digital currencies," *Energy Res. Soc. Sci.*, vol. 44, pp. 399–410, Oct. 2018.
[55] A. de Vries, "Bitcoin's energy consumption is underestimated: A market dynamics approach," *Energy Res. Soc. Sci.*, vol. 70, Dec. 2020, Art. no. 101721.

Oihane Gómez-Carmona received the Ph.D. degree in Internet of Things and artificial intelligence from the University of Deusto, Bilbao, Spain, in 2021.

She is an Associate Researcher with the Deusto Institute of Technology, University of Deusto. Her research focuses on the intersection between IoT and machine learning oriented toward resource-constrained devices and edge computing, as well as the interaction between people and technology-augmented objects.

Diego Casado-Mansilla received the Ph.D. degree in persuasive technologies for sustainability and behavior change from the University of Deusto, Bilbao, Spain, in 2016.

He is an Associate Researcher with the MORElab Research Group, University of Deusto, where he is a Lecturer. He has several relevant publications on the topics of sustainable HCI and Internet of Things. His research interests are focused on sustainable HCI, persuasive and behavior change technologies, sustainable design, and physical interaction with everyday objects.

Diego López-de-Ipiña (Member, IEEE) received the Ph.D. degree in sentient computing from the University of Cambridge, Cambridge, U.K., in 2002.

He is a Full Professor with the University of Deusto, Bilbao, Spain, where he is also a Senior Researcher with DeustoTech and a Principal Researcher with the MORElab Research Group. He has been a principal researcher from DEUSTO part in the EDI, WeLive, SIMPATICO, GREENSOUL and CITY4AGE H2020 Projects, IES Cities CIP Project, MUGGES FP7 Project and the uService ITEA2, CBDP CELTIC and ADAPTA and SABESS National Projects. His main research areas are big data analytics and Internet of Things as enablers for smart cities or to address societal challenges, middleware for mobile and embedded devices, linked data/semantic Web, and social data mining.

Prof. López de Ipiña is a Coordinator of the EDI—European Data Incubator Project, and he acted as a Technical Coordinator of the H2020 WeLive.

Javier García-Zubia (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Deusto, Bilbao, Spain, in 1996.

He is currently a Full Professor with the Faculty of Engineering, University of Deusto, where he also a Leader of the WebLab-Deusto Research Group. His research interests include remote laboratory design, implementation, and evaluation.