Future Generation Computer Systems 154 (2024) 219–234

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

EneA-FL: Energy-aware orchestration for serverless federated learning

Andrea Agiollo [*,1], Paolo Bellavista, Matteo Mendula [1], Andrea Omicini

Department of Computer Science and Engineering, University of Bologna, Italy

ARTICLE INFO

Keywords:
Serverless
Federated learning
Energy management
Internet of things
Resource-constrained learning

ABSTRACT

Federated Learning (FL) represents the de-facto standard paradigm for enabling distributed learning over multiple clients in real-world scenarios. Despite the great strides reached in terms of accuracy and privacy awareness, the real adoption of FL in real-world scenarios, in particular in industrial deployment environments, is still an open thread. This is mainly due to privacy constraints and to the additional complexity stemming from the set of hyperparameters to tune when employing AI techniques on bandwidth-, computing-, and energy-constrained nodes. Motivated by these issues, we focus on scenarios where participating clients are characterised by highly heterogeneous computing capabilities and energy budgets proposing EneA-FL, an innovative scheme for serverless smart energy management. This novel approach dynamically adapts to optimise the training process while fostering seamless interaction between Internet of Things (IoT) devices and edge nodes. In particular, the proposed middleware provides a containerised software module that efficiently manages the interaction of each worker node with the central aggregator. By monitoring local energy budget, computational capabilities, and target accuracy, EneA-FL intelligently takes informed decisions about the inclusion of specific nodes in the subsequent training rounds, effectively balancing the tripartite trade-off between energy consumption, training time, and final accuracy. Finally, in a series of extensive experiments across diverse scenarios, our solution demonstrates impressive results, achieving between *30%* and *60%* lower energy consumption against popular client selection approaches available in the literature while being up to *3.5* times more efficient than standard FL solutions.

1. Introduction

Most recent efforts in the Federated Learning (FL) community focus on original and effective learning paradigms, aiming at optimising model performance and privacy awareness [1–4]. Frameworks typically rely on powerful and homogeneous clients to achieve strong baseline accuracy and low latency. Regrettably, the high-dimensional space of Quality of Service (QoS) constraints describing the set of trade-offs between optimal accuracy, minimal latency, and energy consumption [5,6] is still slackening the adoption of Federated Learning on constrained Internet of Things (IoT) devices. However, there is a wide range of real-world industrial applications that call for FL solutions over constrained devices [7,8]. Indeed, common frameworks impose a homogeneous local setup amongst every client of the federation, hindering the deployability of FL in IoT scenarios, and causing energy waste as well as performance degradation. Ironically, one potential answer to this matter may come from cloud applications. Here, Software as a Service (SaaS) has been easing users to connect to and use cloud-based applications over the Internet in the last few decades. Nowadays, serverless solutions relieve developers from the burden of managing servers and infrastructure-related tasks. In a serverless architecture, developers can focus solely on writing and deploying code without worrying about the underlying infrastructure. Consequently, the academia and the industrial sector widely adopted serverless solutions for their increased flexibility and pluggability. Transparent and fully provider-managed services appear to be the best way to minimise energy costs and speed up the development of real-world applications by allowing insiders to focus on the most challenging phases of software development.

Inspired by the above trends, we introduce EneA-FL as a pioneering serverless Federated Learning (FL) framework, uniquely equipped with a smart energy management module tailored for resource-constrained clients. Conceived to address the energy management challenges inherent in the real-world deployment of FL solutions, this innovative middleware is crafted to facilitate the training and deployment of FL models across a spectrum of heterogeneous Internet of Things (IoT) devices. In particular, the core is an orchestrator that dynamically manages the FL process by shipping to participant nodes a containerised environment capable of monitoring the current status of the hosting machine in terms of: (i)computing and networking capabilities,

* Corresponding author.
E-mail address: andrea.agiollo@unibo.it (A. Agiollo).
[1] Co-first authors.

https://doi.org/10.1016/j.future.2024.01.007
Received 1 August 2023; Received in revised form 21 November 2023; Accepted 6 January 2024
Available online 8 January 2024

(ii)energy budget, and (iii)current accuracy over local samples. As discussed in the paper, our orchestrator dynamically assesses the effort required by each IoT client to fulfil specified QoS. Subsequently, it autonomously applies the most suitable policy, by minimising any intervention by FL users and FL-based application developers. Furthermore, the paper makes a significant novel contribution by showcasing the feasibility of a dynamic approach for coordinating participating nodes in an energy-aware manner. This innovation facilitates easy access for IoT devices to FL capabilities while minimising the need for user or developer intervention. Our automated energy management scheme has been tested over a wide variety of clients and energy requirements while showing its benefits over standard FL approaches. In particular, we tested our solution over CPU- and GPU-enabled microcontrollers with limited computing capabilities and reduced energy budgets [9]. The adaptable container-based approach proposed here has shown to be able to achieve between *30%* and *60%* lower energy consumption against popular client selection approaches available in the literature.

The remainder of the paper is structured as follows. Section 2 describes the current state-of-the-art, highlighting promising directions and lacking efforts in the academic literature. Section 3 illustrates our novel energy modelling scheme, which supports the adoption of containerised applications in fog environments. We depict the architecture of our middleware in Section 4, while Section 5 reports an extensive set of performance results measured by employing different and heterogeneous IoT devices, discussed in Section 6. Finally, Section 7 contains conclusive remarks and directions for future work.

2. Related work

In this section we provide an overview of recent research efforts in the research areas intersecting our proposal—i.e., Serverless containerisation for Machine Learning (ML), and resource management for FL and fog computing scenarios.

Serverless containerisation for Machine Learning: As real-world ML applications continue to expand, there is a growing demand for significantly increased computational power to execute training processes [10]. In this context, cloud resources prove invaluable, enabling the execution of computationally intensive operations that would be impractical for individuals and mid-sized companies. In addition, the combination of serverless paradigm with containerisation approaches enables remarkable scalability at minimal overhead cost [11,12]. While ML-based container orchestration [13–15] has demonstrated the benefits of ML in Serverless computing, the exploration of the applicability of containerised serverless scheme for ML is still pending validation. In this context, several works demonstrate the minimal impact of containerisation on deep learning application performance. Xu et al. [16] results pinpoint how containers have a 0.2%–0.5% overhead compared with host execution time, proving the deep learning containerisation feasibility.

In the distributed ML realm, several works focused on serverless paradigms for FL, such as [17] where the authors illustrate the benefits of a middleware solution that eases the interaction between multiple SaaS cloud providers. Similarly, Singh et al. [18] present a distributed real-time privacy-preserving data analytic solution for smart grid systems based on a Serverless cloud computing FL approach to predict the energy needs of Home Area Networks (HANs). Despite the remarkable relevance of the above-mentioned solutions, none of them have already addressed the feasibility of Serverless computing for FL in a fog environment.

Resource Management in Fog Computing: Extensive analysis of resource management in fog computing environments [19,20] is meticulously explored in [21], where the authors delve deeply into a comprehensive examination of resource management within fog computing settings. The investigation results pinpoint task offloading [22–24] as the predominant resource management approach in the literature. In this context, the need for ensuring transparency [25,26] in offloading decisions for application developers is crucial. Additionally, the heterogeneity of mobile nodes [27] and their energy performance seems still underexplored. This forward-looking perspective aims to promote efficient and developer-friendly offloading strategies in the evolving landscape of mobile computing. Relevantly, the majority of academic efforts tackle offloading strategies by showcasing innovative solutions leveraging simulation tools [28–30] and ignoring QoS factors—i.e., only 18% of the reviewed articles [31–33] take energy considerations into account. These factors represent relevant limitations for enabling real-world deployment of such solutions.

Resource Management in Federated Learning: FL represents the most popular technology for enabling multi-party joint training of ML models. In this context, multiple entities collaborate to locally optimise a shared model by sending their local updates either relying on a central controller or in a fully decentralised fashion. Relying on the local optimisation procedure of the shared model, FL ensures data privacy, while enforcing heavy computational constraints on the federation entities. Therefore, the application of FL to resource-constrained devices – e.g., IoT devices, battery-powered devices, etc. – represents an open research issue [34,35]. In this context, many of the research efforts focus on the identification of effective learning strategies to optimise local model training. Wu et al. [36] propose relying on lower bit-length integers to reduce the computational costs of training and inference models. Similarly, tensor rematerialisation [37], recomputation [38], and efficient architecture strategies [39] have been proposed to reduce memory requirements of Neural Network (NN) training. Although these proposals do not target directly the FL realm, these works focus on resource optimisation during model training and thus can be applied to FL to achieve a more efficient federation scheme. Few works have specifically dealt with the optimisation of resources in the field of FL when the federation is composed of inherently constrained and heterogeneous devices—in terms of either available energy or computational power [40]. The problem of energy-efficient model transmission for FL over wireless communication networks is analysed in [41], where both local computation energy and transmission energy are taken into account, formulating the FL convergence problem as a system energy minimisation problem. The authors in [42] formulate an energy-conscious resource management problem for FL where the federation clients aim to minimise time over a set of energy and communication constraints. Here, the problem is formulated as a Nash equilibrium problem [43], solved in a decentralised fashion. In [44], the optimisation of the local update frequency and the compression ratio of the model to effectively decrease the time required for optimisation is proposed, thus reducing the consumed resources. Cui et al. [45] aims at reducing FL resource usage by optimising their allocation. To this extent, the authors analyse resource block allocation introducing a mixed-integer linear programming strategy to better allocate resource blocks over federation clients.

In this resource efficientisation context, few approaches focus on the client selection process typical of FL. In this context, OORT [46] represents the most popular node selection framework where clients are selected depending on their model “utility”, defined as the potential improvement over the aggregated model. Arouj et al. [47] propose an improved version of OORT – to which we will refer as OORTv2 for the remainder of the manuscript – prioritising clients having higher battery levels to maximise the overall system efficiency. Diversely from these approaches which focus solely on the computation efficiency perspective, [48] analyses the computation and communication efficiency perspective, proposing a *power-of-choice*-based solution. Total time to convergence represents another relevant factor in FL setups, attracting several research efforts such as [49,50]. More in detail, Kim et al. [49] propose to jointly optimise time to convergence and energy consumption in data heterogeneity scenarios, proposing a reinforcement learning client selection algorithm. Similarly, the results

Table 1
Comparison of related work based on their features. Legend: **EA** = Energy awareness, **RE** = Real energy data collection, **NH** = Node Heterogeneity, **C** = Comparison with other solutions, **SA** = Serverless architecture.

Work	EA	RE	NH	C	SA
Grafberger et al. [17]	–	X	–	–	X
He et al. [51]	–	–	–	–	X
Singh et al. [18]	X	–	–	–	X
Yang et al. [41]	X	X	–	–	–
Zaw et al. [42]	X	–	–	–	–
Kim et al. [49]	X	X	X	–	–
Xu et al. [44]	–	X	X	–	–
Cui et al. [45]	–	X	X	X	–
Cho et al. [48]	X	–	–	–	–
Wang et al. [50]	–	X	–	–	–
Lai et al. [46]	–	–	X	X	–
Arouj et al. [47]	X	–	X	X	–
EneA-FL	X	X	X	X	X

in [50] highlight reduced time execution when communication data collection and experiments upscaling over simulated environments are considered.

Although relevant, these approaches do not sufficiently address the academic gap in optimising power consumption within real and heterogeneous environments. While some of the proposed solutions neglect to consider power consumption information entirely, others consider energy optimisation as a direct consequence of training time minimisation, which is not the case when heterogeneous nodes participate in the training process. Ultimately, none of the existing solutions prioritise the concerns of application developers, leading to a lack of seamless integration in a serverless environment. The absence of consideration for developers hinders the smooth and effortless incorporation of these solutions into real-world applications. Therefore, up to our knowledge, there exists no study on the effectiveness of resource management solutions in FL when a set of resource bounds is considered, taking into account heterogeneous resource requirements and availability over multiple clients. Table 1 summarises the limitations of the existing solution and highlights the academic gap that EneA-FL is meant to fill.

3. FL energy consumption modelling in fog deployment environments

One of the objectives of fog computing in several vertical domains of application is to process data in near real-time. Here, a set of edge nodes receives data from IoT devices and performs stream processing with millisecond-grade response time. Given the opportunistic and heterogeneous nature of those scenarios, the unavailability of a node reaching the end of its energy budget is an eventuality to be avoided at any cost. Generally speaking, if we define $E(i)$ as the energy consumption of a fog node i, and consider the canonical FL scenario made up of an arbitrary set of workers of size n and one aggregator node – where ideally, all working nodes presents the same networking and computational capabilities – we can compute the energy consumption of the entire system E_s simply as the sum of the individual contribution of each worker and the aggregator node. Therefore, we can then model E_s as:

$$E_s = \Sigma_i E(i) = n * E_w + E_a,$$

where E_w refers to the energy consumption of a single worker and E_a to the energy consumed for the aggregation process.

By further investigating the contribution of each node in a FL scenario, for a generic worker w the energy consumption E_w is directly proportional to the local model complexity – referred to as $M_c(w)$ – and to the size of the local dataset—referred to as $S_d(w)$. Given that in a FL scenario all the participating nodes share the same model architecture, we have that:

$$M_c(w_1) = M_c(w_2) = \cdots = M_c(w_{n-1}) = M_c(w_n) = M_c$$

Meanwhile, as far as the aggregator node a is concerned, the computation E_a is usually an average-like operation with a limited corresponding computational cost. However, this operation can became expensive with a high number of tensor to be averaged. This consideration lead us to the conclusion that E_a is directly proportional to the number of workers n.

The energy modelling scheme described above would be incomplete if the relationship between energy, latency, and accuracy would not be made explicit. In fact, the main priority of a FL pipeline is to provide the best accuracy as possible while minimising the overall latency. While the better accuracy Acc usually corresponds to a higher M_c, the minimisation of the overall latency ΔLat is beneficial to energy consumption, too. The optimisation of these constrains would be easy to solve in an ideal scenario with a low level of heterogeneity among participating working nodes. Theoretically speaking:

$$\begin{cases} \min(\Delta Lat) \mathrel{\hat{=}} \min(M_c) \\ \max(Acc) \mathrel{\hat{=}} \max(M_c) \\ \min(E_s) \mathrel{\hat{=}} \min(M_c) \end{cases} \tag{1}$$

Then, the minimisation of M_c would be beneficial for E_w, E_a and ΔLat. In other words, the one between Acc and M_c would be the only trade-off to solve to minimise energy consumption while satisfying QoS requirements.

Unfortunately, the erratic nature of fog environment brings into play an higher-dimensional space of possible solutions. Given that each worker node may be involved in more than one task and that it may have a dynamic percentage of bandwidth at its disposal over time, the selection of a specific node during the aggregation phase may be tricky and negatively impactful not only in terms of Acc and ΔLat, but also for E_s. For this reason, the dynamic selection of workers over the time is a primary task to address in order to apply FL in real-world fog scenarios.

4. The EneA-FLserverless middleware

In this section, we present the architecture of our novel serverless middleware for FL in fog environments, namely EneA-FL, shown in Fig. 1. Inspired by the emerging Cloud Continuum paradigm, EneA-FL presents the first middleware capable of bringing together the best characteristics of serverless and fog computing, showcasing its applicability to FL settings with highly dynamic and heterogeneous devices.

4.1. Serverless computing and energy awareness in constrained scenarios

The harmonic combination of microservice architecture and energy awareness emerges as a natural consequence of applying the serverless paradigm in fog computing scenarios. The microservice architecture, with its modular and decentralised approach, enables the development of flexible and scalable applications by breaking down complex functionalities into smaller independent services, by easing software portability over heterogeneous IoT devices at the same time. Meanwhile, energy awareness focuses on optimising resource consumption and power utilisation to achieve energy efficiency in constrained computing environments. When integrated into fog computing, which extends cloud services to the edge of the network, the serverless paradigm brings its on-demand execution and resource management capabilities. This allows applications to leverage microservices while efficiently utilising resources and minimising energy consumption in the edge and fog nodes. By combining these elements, fog scenarios can harness the benefits of microservice-based application development while maintaining energy-conscious operations. This fusion facilitates the creation of responsive, adaptable, and energy-efficient systems, making it an advantageous approach for deploying applications in dynamic and resource-constrained edge environments.

The synergy between microservice-oriented architecture and energy awareness for serverless fog computing represents a significant step towards building sustainable and high-performance applications at the edge of the network. In this context, EneA-FL is an original and relevant contribution, by providing FL developers with the first framework supporting the hybrid composition of serverless and fog computing by taking into account the resources consumed by participating nodes. By bridging the gap between serverless computing and fog scenarios, EneA-FL capitalises on the benefits of on-demand execution and resource optimisation, while extending these advantages to the edge of the network.

4.2. EneA-FL architecture

EneA-FL consists of three main modules:

Energon Prometheus[2] is a well-established open-source systems monitoring and alerting toolkit originally built at SoundCloud [52]. Inside EneA-FL, Energon is the Prometheus-compliant exporter for IoT and edge devices that keeps track of each participating device status in a completely transparent way. It is shipped to participating nodes as a container that passively collects the energy metrics of the host, independently of the local operating system and offers to the aggregator an endpoint for polling energy and network metrics It scrapes a wide set of Linux-based microcontrollers by reading specific registries at the operating system level Energon is published as a Pypi package[3] to help the community track down the system metrics of edge device transparently

Furcifer A novel container orchestrator that handles the communication between participants nodes by offering an overlay network [53,54] to each node inside the cluster This enables peer-to-peer communication between participating nodes and corresponding containers over a virtual network managed through a Docker DNS interface In addition, it provides each participating node with a kernel-compliant container-based application

Magister A policy manager that takes care of the aggregation process by choosing the aggregation policy and selecting the participant clients for each aggregation round It is also in charge of deciding when the learning process is completed depending on the grade of satisfaction of the specified QoS requirements for the FL application

4.2.1. Energon for transparent energy awareness

Energon is a modular monitoring tool for IoT and edge devices. It keeps track of an extensible set of system metrics about energy consumption, network channel quality, and resource utilisation, among others. Collected metrics are compliant with the Prometheus exporting standard, which was recognised as graduated project maturity level in 2016 by Cloud Native Computing Foundation, the open-source vendor-neutral hub of cloud-native computing. Diagnostic information can be obtained by HTTP requests to the*/metrics* endpoint on the IoT device. This allows both scanning with application-dependent business logic and a smooth interaction with the Prometheus ecosystem. The ubiquitous nature of Energon allows the user to monitor critical metrics and to make real-time decisions at the application level, without generating any additional overhead for the constrained devices. When it comes to system monitoring, one of the primary objectives is to minimise the additional operations necessary to collect the desired metrics while the target applications are running. Energon has been meticulously designed to ensure complete isolation from the rest of the system. It operates independently and does not require any interaction

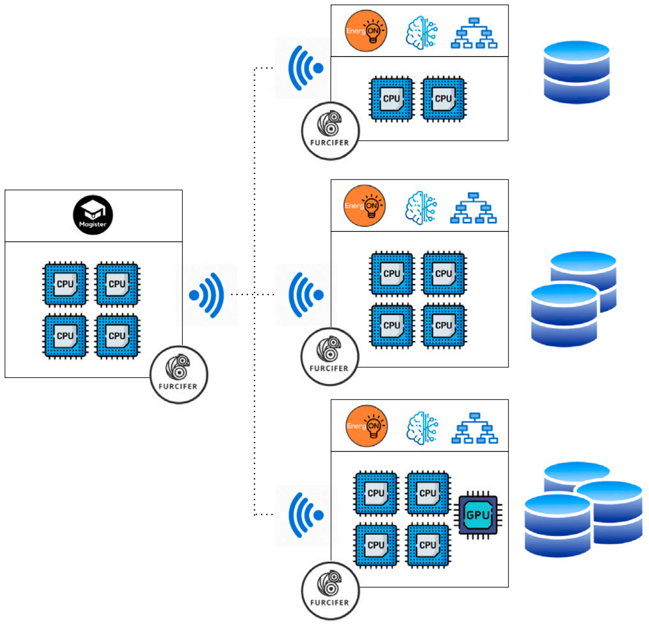


Fig. 1. Serverless middleware architecture.

with the running applications, thereby eliminating any potential interference or performance overhead caused by monitoring processes. Running as a separate process allows Energon to efficiently collect the desired metrics and perform monitoring operations without being tightly coupled to the application's execution and its business logic. By adopting this approach, Energon efficiently and seamlessly gathers essential metrics without impacting the performance and behaviour of the monitored applications, making it an effective and non-intrusive solution for system monitoring tasks. This helps developers focus solely on the development of their applications without the need to worry about logging the device's state for later historical analysis or real-time decision-making.

In addition to raw data monitoring and exposition, Energon can be customised to send an event when a specific condition is met. Inside our EneA-FL middleware, Energon plays the central role of keeping the orchestrator updated about the current state of the monitored nodes, thus allowing the policy manager (i.e., Magister) to select the best workers available in terms of residual energy, instantaneous power consumption, and peer-to-peer communication quality. About the querying interface, Energon wraps PromQL, the functional expression language defined by Prometheus, with easier high-level REST APIs. Those JSON-based endpoints can be further customised depending on QoS requirements. Scraped metrics can be stored locally on the orchestrator side, as done inside EneA-FL, or they can be saved on a separate database for later use—e.g., time series analysis for designing new better policies. All metrics are stored as time series data identified by a metric name and a set of key–value pairs. Sharding and federation are also possible with minimal additional settings.

4.2.2. Furcifer : Container orchestrator for IoT devices

Furcifer is a centralised microservice manager, specialised for constrained devices and enabling communication between participant nodes. It is available in multiple versions through a set of containers for different operating systems and architectures. Furcifer is meant to fill the gap between potential policy-oriented adaptation and the heterogeneous nature of fog computing, by exploiting container orientation – and not virtual machines –, largely accepted as more suitable for these deployment environments. The introduction of an additional abstraction layer offered by containers is justified by the minimal impact on system resources and the need for higher flexibility for context

[2] https://github.com/prometheus/prometheus.
[3] https://pypi.org/project/energon-prometheus-exporter/.

adaptation purposes. In particular, when focusing on FL deployment over edge devices, we have to consider model drifting as a critical circumstance where model performance drops in an unpredicted manner. When model drifting is detected, it is very likely that a new model architecture has to be deployed on all edge devices. In a traditional setup, where all participating nodes are executing their local training at the OS level without any kind of containerisation, a manual deployment of the updated model will have to take place, with additional effort and substantial impracticality in real-world scenarios—e.g., required intervention for each participant IoT device. On the other hand, if a new model architecture has to be deployed in a containerised environment the only operation required by workers is to pull a new image from the container registry offered by Furcifer. This wraps and extends Harbor,[4] a well-known open-source container registry supporting Kubernetes-based applications, for fog scenarios. In addition, since the majority of the dependencies are likely to be unchanged, only the last layer of the image will be downloaded, minimising at the same time maintenance operations and bandwidth utilisation. When a container image is built, the platform builder attempts to reuse layers from earlier builds and if a layer of an image is unchanged, then the builder picks it up from the build cache without any additional download. As a consequence, even if the container occupies more memory compared with bare metal solutions, only the device initialisation phase is affected; while real-time adaptations minimally impact the context switch latency.

Furcifer use of container images significantly simplifies the deployment and scaling of applications. Once an updated container image is available, it can be easily distributed to all participating edge devices through the container registry, ensuring that the latest version of the application or model architecture is seamlessly deployed across the network. This streamlined process minimises manual intervention and reduces operational overhead, making it feasible to manage a large fleet of IoT devices efficiently. Furthermore, Furcifer overcomes the lack of hardware acceleration support on constrained devices by integrating *NVIDIA Engine Runtime* for GPU-enabled IoT devices in a transparent way. The orchestrator checks automatically the availability of hardware acceleration and reserves the GPU for the FL training process before it takes place. This feature has been successfully tested inside EneA-FL on NVIDIA Jetson family boards.

In terms of storage utilisation, containers occupy a relatively larger amount of additional space compared to OS-level applications. However, it is crucial to note that the business-logic application and ML model utilise less than 5% of the total space, while the majority of storage is dedicated to user libraries. This efficient distribution means that when deploying a new model or communication strategy, only the last layer of the image needs to be shipped to the worker node, equivalent to the size of an OS-level application. Consequently, the additional storage introduced by containerisation significantly impacts initialisation time, but once the application is running, it does not exacerbate overall latency. This highlights the advantage of containers in efficiently managing application dependencies and minimising the impact on run-time performance once the containerised application is operational. In addition to providing support for real-time adaptation scenarios and dynamic resource allocation, Furcifer offers an essential advantage in terms of security and isolation. Thanks to containerisation, each service and application running on the constrained devices is encapsulated within its own container, creating a boundary that restricts its access to system resources. Isolation ensures that if one container is compromised, the security of other containers and the host system remains intact. This level of security is particularly crucial in edge computing and FL environments, known to be susceptible to attacks [2,4,55]. In addition, defining the communication interface at the container level provides an additional layer of security in the system. With this setup, any potential malicious node attempting to communicate with the central aggregator would face significant hurdles. Such a node would need to compromise and flawlessly replicate an existing image used in the container to establish communication.

4 https://goharbor.io/.

4.2.3. *Magister for fog-oriented context switch decision making*

The worker selection problem in FL has been approached from various perspectives, including faster convergence [56–58], higher trust level [59–61], and energy-awareness [62,63]. However, selecting the most suitable workers in terms of both energy consumption and training speed becomes challenging in scenarios where multiple heterogeneous devices interact and multiple tasks must be executed simultaneously. Complexity arises from the diverse capabilities and resource constraints of the devices involved, making it crucial to develop innovative and efficient approaches to address the worker selection dilemma in such dynamic and diverse environments. While the identification of the most efficient device can be done in a static and self-evident way by considering the amount of FLOPS per Watt on each device, a more dynamic adaptation is required when additional constraints take place. For example, the most efficient device may not be available due to the limited battery duration or to the parallel execution of a different task with higher priority. In addition, network channel quality can also play a central role when dealing with mobile devices. For these reasons, even though in a controlled environment a static energy-preserving policy may be suitable, this is not applicable when moving to real-world scenarios. A higher-dimensional space of constraints has to be taken into account to save as much energy as possible while meeting QoS requirements. Inside EneA-FL, Magister is the module of the container orchestrator in charge of optimising the clients selection policy depending on the state of each worker in terms of system metrics and QoS satisfaction. In particular, Magister takes into account the consumed energy of participating workers, the time required to perform the local training procedure and the accuracy improvement compared with the previous training epochs. Communication is not taken into account at this level.

At the beginning of each federation round, Magister collects the clients' resource usage from Energon – using Furcifer – and selects clients accordingly. The local training procedure is not impacted by Magister, which upon the reception of local updates from the selected clients decides whether to keep the distributed training process alive or to stop it—depending on the achieved QoS metrics. Therefore, the global model aggregation process is also not affected by our solution. As a result, Magister represents a flexible client selection component which can be integrated with any custom training and aggregation mechanism for FL. To effectively identify the Magister smart selection process, here we analyse how the worker selection process affects the consumed energy, execution time and accuracy improvements. First, we define the workers available to the federation process a $\mathcal{W} = \{w_1, w_2, \dots, w_N\}$, where N represents the total number of workers that compose the federation. Out of these N workers only a subset $\mathcal{W}_S \in \mathcal{W}$ is selected by the aggregating entity for each round of the optimisation process. The selection function used to identify $\mathcal{W}_S$ is defined as $\mathcal{S}\{\mathcal{W}\}$, and is usually considered to be a simple random selection process in most FL setups. Magister's objective is to identify some novel selection procedure $\mathcal{S}^*$ minimising the amount of energy and latency required by the federation process while maintaining the performance of the aggregated model untouched. To this extent, we first consider the selection function $\mathcal{S}$ to be dependent on the history of the federation process, accounting for smart selection of highly impactful nodes and disregarding unreliable workers.

To identify the optimal selection function, we need to take into account the dependency between the achieved performance of the aggregated global model and the amount of energy and time spent by the FL system to reach this global optimum. To this end, we here define the neural network model trained by worker w_i at the tth optimisation step of the federation process as $\mathcal{N}_i^{(t)}$. Consequently, the amount of energy used to obtained $\mathcal{N}_i^{(t)}$ via local computation is defined as $E_c(\mathcal{N}_i^{(t)})$. The time it takes for the optimisation process in worker w_i to compute $\mathcal{N}_i^{(t)}$ is written as $\tau(\mathcal{N}_i^{(t)})$. Meanwhile, at the aggregator node, the neural network model computed at the end of the tth federation step is defined as $\mathcal{N}_a^{(t)} = \mathcal{A}\{\mathcal{N}_i^{(t)}\ \forall i \in \mathcal{W}_s^{(t)}\}$, where $\mathcal{A}$ represents a custom aggregation function used to compute the global model from the local updates. FedAvg [64], where $\mathcal{A}$ corresponds to the average weights of the local model, represents the most popular aggregation solution thanks to its simplicity of competitor solutions [65,66]. If we consider relying on a test set of examples to compute the goodness of the obtained aggregated global model, we can define its performance – e.g., accuracy, f1-score, etc. – as $\mathcal{P}(\mathcal{N}_a^{(t)})$. The overall aim of any FL process is to increase the performance of the aggregated model $\mathcal{P}(\mathcal{N}_a^{(t)})$ at each federation step t, so that $\mathcal{P}(\mathcal{N}_a^{(t+1)}) \geq \mathcal{P}(\mathcal{N}_a^{(t)})$ and the global optimum is achieved at the end of the federated optimisation process.

To account for smart selection of the federation workers, we here consider relying on a selection function that aims at maximising the performance of the aggregated model, while minimising the energy and time spent. Therefore, we would be ideally able to define a new selection function $\mathcal{S}$ that selects the next set of workers $\mathcal{W}_S^{(t+1)}$, such that:

$$\mathcal{W}_S^{(t+1)} \; s.t. \begin{cases} \max\left(\mathcal{P}\left(\mathcal{N}_a^{(t+1)}\right)\right) \\ \min\left(\sum_{\mathcal{W}_S^{(t+1)}} E\left(\mathcal{N}_i^{(t+1)}\right)\right) \\ \min\left(\sum_{\mathcal{W}_S^{(t+1)}} \tau\left(\mathcal{N}_i^{(t+1)}\right)\right) \end{cases} \tag{2}$$

However, while desirable, such a setup cannot be solved directly, as the nodes that mostly affect the most the computation of the aggregated model $\mathcal{N}_a^{(t+1)}$ cannot be known *a priori*. Therefore, we here rely on the assumption that the setup can be computed *a posteriori*. Thus, we aim at identifying the set of nodes that affect mostly positively the performance of the aggregated model for previous steps $t, t-1, \dots, 1$, while consuming less resources – i.e., energy and time –, and rely on these nodes also for aggregation step $t+1$. Therefore, the selection process becomes a combinatorial search problem, where we aim at identifying the subset of best workers $\hat{\mathcal{W}}_S^{(t)}$, such that their combination achieves high performance while keeping both energy and time requirements under control.

As it represents a combinatorial optimisation problem, finding the optimal set of workers $\hat{\mathcal{W}}_S^{(t)}$ to maximise performance and minimise resources is not scalable when the number of workers increases. Therefore, relying on such an optimal procedure would lead to huge computational waste from the aggregator node end. While we aim at minimising the amount of resources spent at workers level, shifting the computational burden to the aggregator node would not represent a feasible solution. Thus, we here consider relying on a simplified approach that does not require the solution of the combinatorial optimisation problem. To this end, we here define an effectiveness measure aiming at identifying how much the local model update from a worker w_i impacts the performance and resource consumption of the whole federation setup.

Accordingly, we define the effectiveness metric as:

$$\mathcal{E} = \alpha \cdot \frac{E\left(\mathcal{N}_i^{(t)}\right)}{\max\{E^{(t)}\}} + (1-\alpha) \cdot \frac{\tau\left(\mathcal{N}_i^{(t)}\right)}{max\{\tau^{(t)}\}} - \beta \cdot \Delta\left(\mathcal{P}\left(\mathcal{N}_a^{(t)}\right), \mathcal{P}\left(\mathcal{N}_{a\ominus i}^{(t)}\right)\right), \tag{3}$$

where $\max\{E^{(t)}\}$ represents the maximum energy spent by any worker at step t, namely $\max\{E^{(t)}\} = \max\{E\left(\mathcal{N}_j^{(t)}\right) \forall j\}$. Similarly, $\max\{\tau^{(t)}\}$ represents the maximum time taken by any worker at step t, namely $\max\{\tau^{(t)}\} = \max\{\tau\left(\mathcal{N}_j^{(t)}\right) \forall j\}$. Meanwhile, Δ represents the difference in performance between the aggregated model $\mathcal{N}_a^{(t)}$ and the model $\mathcal{N}_{a\ominus i}^{(t)}$ obtained aggregating all workers updates except w_i. Finally, α and β represent the hyperparameters that identify the trade-off between the relevance of energy consumption, time requirements, and the performance improvement achieved. Intuitively, a local worker w_i whose model has a high impact and is obtained by spending little resources ensures a small score. On the other hand, workers whose models have little-to-no impact on the model performance and/or are obtained consuming a vast amount of resources result in high scores. To account for devices that are not capable of sending an update to the aggregation entity, we set a handicap value h to the effectiveness metric of those devices that do not guarantee an update when selected, namely:

$$\mathcal{E}\left(w_i^{(t)}\right) = \begin{cases} \mathcal{E} & \text{if received update} \\ h & \text{otherwise.} \end{cases} \tag{4}$$

Finally, to take into account the reputation history of workers, we consider measuring the reputation score of each worker as its average effectiveness score, namely:

$$\mathcal{R}\left(w_i^{(t)}\right) = \frac{\sum_{l=1}^{t} \mathcal{E}\left(w_i^{(l)}\right)}{t}. \tag{5}$$

Considering the reputation score of Eq. (5), we can now redefine the problem of identifying optimal workers as the process of identifying

$$\hat{\mathcal{W}}_S^{(t)} \; s.t. \; \min\left(\mathcal{R}\left(w_i^{(t)}\right)\right). \tag{6}$$

Therefore, in our novel workers selection process, the workers selected to run the optimisation procedure for the next aggregation step $t+1$ are

$$\mathcal{W}_S^{(t+1)} = \left\{\hat{\mathcal{W}}_S^{(t)} \; s.t. \; \min\left(\mathcal{R}\left(w_i^{(t)}\right)\right) \oplus \mathcal{S}_r\left(\mathcal{W} - \hat{\mathcal{W}}_S^{(t)}\right)\right\}, \tag{7}$$

where $\hat{\mathcal{W}}_S^{(t)}$ represents the set workers with optimal performance from step t – to account for incentivisation of highly performing nodes – and $\mathcal{S}_r\left(\mathcal{W} - \hat{\mathcal{W}}_S^{(t)}\right)$ represents a random sampling – i.e., $\mathcal{S}_r$ – of the remaining non-optimal nodes—i.e., $\mathcal{W} - \hat{\mathcal{W}}_S^{(t)}$. The selection of a set of random non-optimal nodes is necessary for the federation process to take into account nodes that were not present in previous aggregation procedures, which may still impact positively the global model. The cardinality of the set of optimal nodes and randomly selected ones accounts for the trade-off between static behaviour over time and workers coverage. Thus, we add a third hyperparameter value k that balances the cardinality of optimal nodes and randomly selected ones. In particular, if we identify with O the cardinality of $\hat{\mathcal{W}}_S^{(t)}$ – i.e., the number of optimal nodes selected from one iteration to the next – and with R the cardinality of $\mathcal{S}_r\left(\mathcal{W} - \hat{\mathcal{W}}_S^{(t)}\right)$—i.e., the randomly sampled non-optimal nodes, and with n_r the number of workers to be selected at each round, we obtain $O = \lfloor k \cdot n_r \rfloor$ and $R = \lceil (1-k) \cdot n_r \rceil$. Taking into account energy consumption, time, and reached performance level of each client in the federation – Eq. (3) –, EneA-FL handles device heterogeneity in terms of computational capabilities, computational efficiency, and data availability.

5. Experimentation testbed and experiment preliminaries

As already stated, our primary objective is to address the pressing need for more energy-efficient and sustainable machine learning models, especially in the era of ubiquitous data and resource constraints. By leveraging the collaborative power of FL, we aim at demonstrating significant energy savings without compromising model performance.

5.1. FL training process and model complexity

The experiments are conducted on a real networking testbed consisting of heterogeneous edge devices, which mimics a real-world deployment environment with diverse computational capabilities. We use representative datasets from LEAF [67], which includes different use case data coming from Computer Vision and Natural Language Processing areas. From LEAF we select two datasets: MNIST and Sent140 to test the feasibility of our energy-aware Federated Learning selection on multiple tasks. While MNIST is the distributed version of the well-known MNIST dataset for image classification, Sent140 consists of a corpus of 1,600,000 tweets extracted using the Twitter API for sentiment analysis. Finally, we also employ the N-BaIoT dataset [68]

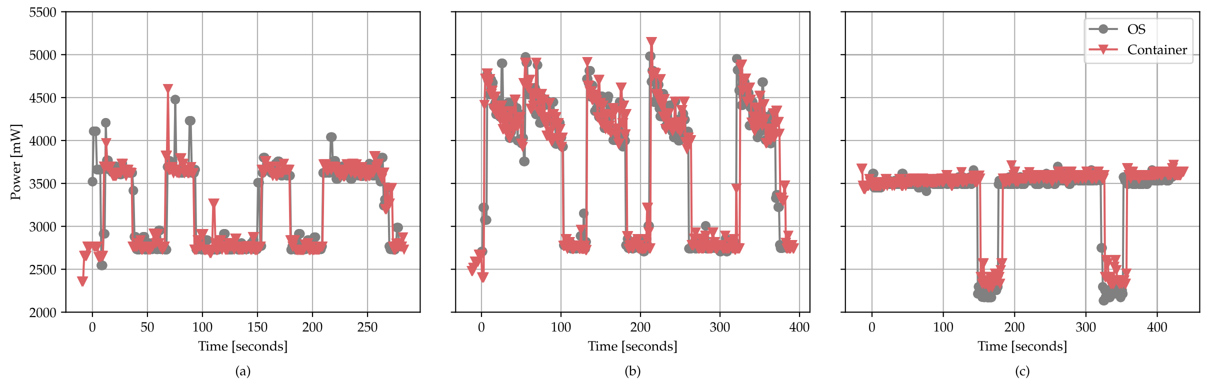


Fig. 2. Time execution and energy consumption comparison on GPU enabled Jetson Nano over 5 training epochs. (a) On Sent140 dataset. (b) On MNIST dataset. (c) On N-BaIoT dataset.

Table 2
Complexity of leveraged NN models.

	MAC	n params
Sent140	16.106 M	2.006 M
MNIST	5.962 M	2.278 M
N-BaIoT	4.384 K	4.491 K

to include a dataset containing realistic traffic data gathered from 9 commercial IoT devices authentically infected by Mirai and BASHLITE. Here, the malicious traffic is divided into 10 different attack classes (e.g., network scanning and firmware) plus 1 benign class. On all datasets, we implement a small neural network composed either of a few blocks of convolution operations or linear ones. The corresponding model complexity is evaluated in terms of both MAC (Multiply-Accumulate Operations) and the total number of parameters, reported in Table 2. The overall model complexity of the model for image classification on MNIST is more than three times the one for sentiment analysis over Sent140 and 1000 times the one for attack identification in N-BaIoT.

5.2. Container vs. operating system FL training execution

Before delving into finer details, we perform a comparison between the containerised FL training application and the execution at operating system level. We execute EneA-FL with and without containerisation to estimate the additional overhead brought by a higher level of abstraction. The training process consists of five local epochs for each dataset where both os-level and container training execution follow the same node selection policy. For this reason, the device is selected or ignored depending on the current state of the federation. This unveils noteworthy similarities in their energy consumption, with a minor difference of about 5% in terms of training time. As can be seen in Fig. 2, both executions demonstrate comparable energy usage, indicating that the containerisation process does not significantly impact overall power consumption during training tasks. However, the containerised environment exhibits a slight delay, with training times being approximately 1%–2% slower compared to the operating system level environment. As it can be seen, the most powerful device, the Jetson AGX Orin, exhibits significant variability and dispersion in its values; while the two most efficient devices, the Jetson Nano and the Jetson Xavier tend to manifest a consistent and steady behaviour, in particular when equipped with hardware acceleration. Generally speaking, over all the devices considered, the usage of GPU seemingly leads to lower data sparsity and a more predictable trend. The overall marginal disparity in terms of startup required time proves the minimal overhead versus the benefits of GPU-enabled containerisation. The improved portability, isolation, and ease of deployment remain highly advantageous, making it a viable and efficient choice for FL training tasks in fog environments.

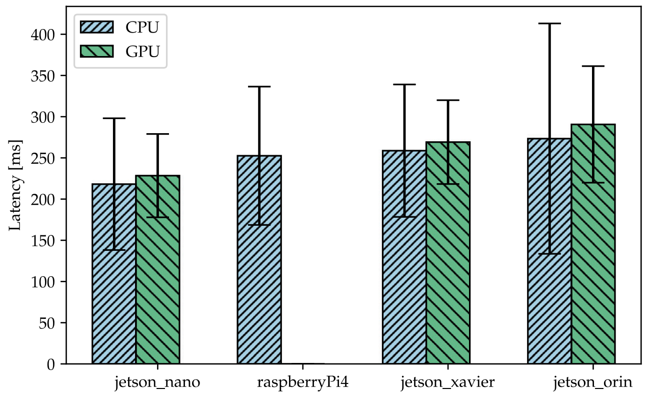


Fig. 3. Time startup comparison of CPU containers vs. GPU-enabled containers.

5.3. Container startup latency assessment

In the dynamic landscape of IoT and serverless architecture, a crucial aspect of consideration revolves around the comparison of container startup latencies when utilising IoT devices with and without GPU hardware acceleration. Serverless computing, which enables on-demand execution of functions without the need for server management, is particularly attractive in IoT environments where data is generated at the network's edge. Containers, acting as self-contained units of application code and dependencies, can significantly impact the responsiveness of IoT applications during initialisation and deployment. By assessing and contrasting the startup latencies with and without GPU hardware acceleration, we gain valuable insights into the potential performance gains and resource optimisation for serverless architectures in the context of IoT applications. This section analyses the minimal additional overhead given by containerisation during training execution. In particular, we evaluated the startup latency for all the devices taking into account over 100 startup cycles.

The data presented in Fig. 3 demonstrates the startup latencies of various configured devices, revealing that the presence of hardware acceleration does not result in significantly higher delays. On average, the additional latency incurred with hardware acceleration is merely 5%–10%. Moreover, across all devices, the observed latencies range between 200 ms and 300 ms, underscoring the minimal overhead introduced by containerisation techniques on the overall training execution. These findings highlight the efficiency and effectiveness of utilising hardware-accelerated containers, as they offer near-instantaneous startup times while providing substantial benefits in terms of flexibility and modularity during training processes.

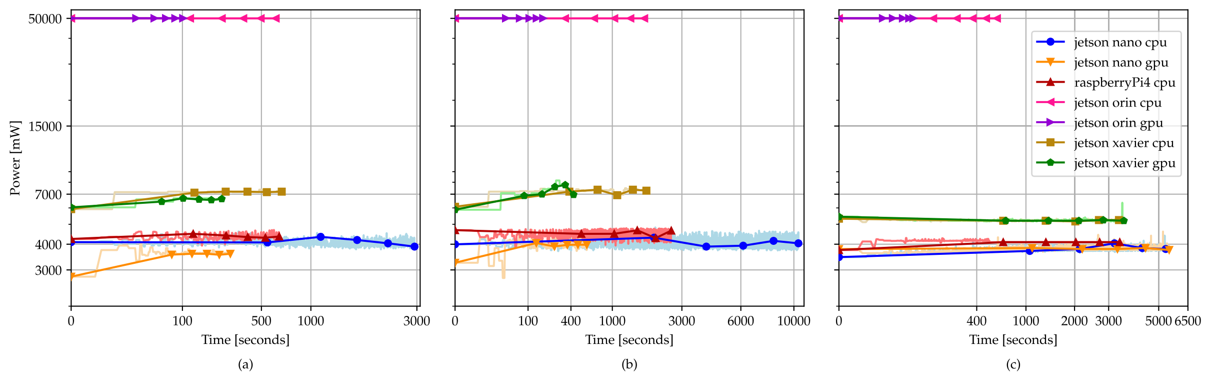


Fig. 4. Time execution and energy consumption on considered devices over 5 training epochs. (a) On Sent140 dataset. (b) On MNIST dataset. (c) On N-BaIoT dataset.

5.4. Energy consumption and training execution time on IoT devices

To evaluate the performance and the effectiveness of energy optimisation, we used the following metrics: model accuracy, communication rounds and clock time required for convergence, and total energy consumption. We first evaluate the energy consumption of a wide range of IoT devices commonly used in fog and edge scenarios. In particular, we measure the energy consumption of our models on a Raspberry Pi model 4, a Jetson Nano developer kit, a Jetson Xavier NX board, and a Jetson AGX Orin developer kit. In addition, each of the Jetson boards has been tested with and without GPU support to check the energy cost of hardware acceleration in IoT training processes. Fig. 4 illustrates the experiments performed to measure the energy consumption of all devices tested in five training epochs. The Jetson AGX Orin developer kit is the fastest, but it is also the one with the higher instantaneous power consumption. On the contrary, the slowest device is the Jetson Nano without GPU, while the hardware-accelerated version of the device appears the be the best compromise in terms of energy consumption and training time.

Surprisingly, the notion of utilising low-resource devices, like the well-known Raspberry Pi model 4, proves to be unfavourable in terms of both overall energy consumption and training execution time. When considering the total energy cost over the whole training, employing a higher-powered device may lead to lower energy consumption. Additionally, the experiments underscore the remarkable reduction in training execution time achieved with higher-tier devices, exemplified by the Jetson AGX Orin, albeit at the expense of consuming approximately 10 times more energy compared to other devices. On the other hand, the Raspberry Pi 4 requires over five times more time for training execution compared to the majority of other constrained devices. Notably, in the first two datasets, the difference in training time between GPU-enabled devices and their CPU-limited versions is significant. However, on the N-BaIoT dataset, this difference is minimal. As expected, in this case, the lower complexity of the model does not benefit significantly from parallel computation. led by hardware-accelerated resources.

Fig. 5 showcases the energy efficiency of the examined devices, determined by computing the amount of Joules using Simpson's rule for numerical approximation integration [69]. This method allows us to precisely evaluate the energy consumption of each device and compare their efficiency in performing the given task. By employing Simpson's rule, we gain valuable insights into the energy performance of the devices, providing a comprehensive understanding of their capabilities in optimising power utilisation during the integration process. According to the results, the Jetson AGX Orin emerges as the least efficient device in terms of energy consumption, indicating higher energy usage during the integration process. On the other hand, the Jetson Nano and Jetson Xavier stand out as more energy-efficient options, as they exhibit lower power consumption relative to their execution times.

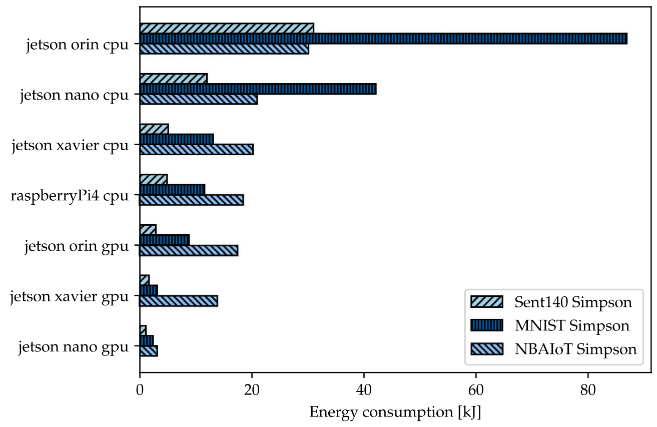


Fig. 5. Ranking of overall energy consumption over considered devices.

6. EneA-FL performance results and discussion

In this section, we present the results obtained during the testing phase of EneA-FL framework. More in detail, we first describe the setup used for the analysis of our framework in Section 6.1. In Section 6.2 we perform an ablation study to identify EneA-FL best hyperparameters. We then study the performance of the EneA-FL framework in terms of reached accuracy when a set of resources budgets are considered (see Section 6.3). Sections 6.4 and 6.5 analyse respectively the impact of the number of selected devices per round and the inactive devices on the performance of EneA-FL. Finally, in Section 6.6 we study the flexibility of EneA-FL over variations of the distribution of device types in the federation.

6.1. Experiments setup

To test the proposed EneA-FL framework we implement a federated learning simulation tool that takes into account the testbed results obtained in Section 5 to emulate the deployment of containerised workers on each of the seven selected devices. More in detail, we implement the proposed framework using PyTorch [70] for the definition of models and their learning process and rely on FedAvg [64] as the aggregation algorithm during the FL training process. Through all our experiments, we consider a federation network composed of 100 different worker and for each worker we identify the probability of it being one of the seven device types considered in Section 5. We refer to the distribution of these probabilities as the distribution of device types and, apart from the experiments in Section 6.6, we build the federation using a uniform distribution of device types. In particular, since we consider 7 different type of devices – see Section 5.4 – we deploy a federation composed by

14.29% of each device type. Thus we obtain a setup where each device type covers one-seventh of the whole federation hardware.

The energy consumption and execution time of the local training process of each device is emulated by the worker belonging to the federation to keep track of the total energy consumption and total execution time. More in detail, for each training step of the worker local training process, we perform a sampling from the normal distribution of the energy consumption and time requirements that characterise the device type at hand. The characteristic normal distribution is extracted from the testbed measurements obtained in Section 5, thus representing faithfully the real-world energy consumption and latency of the selected devices.

The implemented framework also allows for a flexible definition of several additional options, such as device type distribution, device lifetime, data distribution, QoS target definition and many more. The device lifetime is modelled as an exponential random variable that models the number of available federation rounds and is used in Section 6.5 to assess the impact of device discharge on EneA-FL. Meanwhile, QoS targets definition supports various types of resource budgets – investigated in Section 6.3 – and target performance. Finally, if not diversely specified, throughout our experiments we consider uniformly distributed data samples over all workers belonging to the federation. For maximum results reproducibility, the developed code and documentation for all the experiments presented in this article are published on a dedicated EneA-FL GitHub repository.[5]

6.2. Ablation study

EneA-FL relies on its Magister component for optimisation of the device selection process throughout federation learning. In this context, the energy-aware policy defined in Section 4.2.3 takes into account a mixture of energy consumption, latency and achieved performance to promote energy-effective workers. To balance these three components EneA-FL relies on three hyperparameters to select the devices of each federation round, namely: (i)α, balancing energy consumed and execution time; (ii)β, to account for local models' accuracy; and (iii)k, adding a randomness component to the selection process. Given the relevance of these hyperparameters in EneA-FL we here propose an ablation study aiming to identify the best Magister setup.

6.2.1. α And β values

The energy effective metric proposed in Eq. (3) relies on α and β hyperparameters to weigh the relevance of minimal power consumption, training execution time and achieved accuracy. By leveraging the values of α and β, EneA-FL can balance the trade-off between energy consumption over training execution time. Therefore, it represents a significant aspect of the identification of the best α and β setup for running EneA-FL. Indeed, setting these hyperparameters is not a straightforward process, as α is the component setting a trade-off between energy and time, while β represents the parameter tuning the relevance of the achieved accuracy. More in detail, higher values of α define an optimisation process where the energy is considered more valuable than the execution time of the learning step, thus giving higher priority to very efficient – possibly slow – devices. Meanwhile, smaller α prioritises raw optimisation speed over energy efficiency, valuing faster devices that may consume more energy. On the other hand, β represents ideally the relevance given to the reached accuracy of local models. Therefore, higher β allows Magister to promote devices whose model reaches higher accuracy improvements, without regarding their energy consumption and latency. Meanwhile, smaller β allows Magister to focus solely on optimisation energy and latency, disregarding the reached accuracy.

[5] https://github.com/AndAgio/EneA-FL.

Table 3
Comparison of energy consumption, time execution and rounds required to converge on MNIST dataset between different worker selection approaches and the best and worst (α, β) EneA-FL setups. The EneA-FL best setup is obtained for $\alpha = 0.6$, $\beta = 40$, while its worst setup is obtained for $\alpha = 0$ and $\beta = 100$. For each metric, we highlight in **green** the best approach.

Approach	Energy	Time	Rounds
Standard FL	4.6 MJ	28.5 h	9.7
OORT [46]	1.8 MJ	13.2 h	11.2
OORTv2 [47]	2.2 MJ	13.8 h	13.8
EneA-FL worst	2.1 MJ	14.1 h	10.7
EneA-FL best	1.3 MJ	12.7 h	10.8

To analyse the impact of α and β values on the federation optimisation we let their values vary between 0 and 1, and 0 and 100 respectively. We perform 10 federation optimisation experiments for each α and β combination, setting the number of federation rounds to 30. Training is performed over the MNIST dataset. For each experiment, we keep track of the (i) overall energy consumption; (ii) total execution time; and (iii) number of rounds required to converge. Here, convergence is considered to be achieved when the global model achieves 97% of accuracy on the test set.

Fig. 6 shows the average values for energy, time and convergence achieved over the 10 experiments of each α and β setup. More in detail, Fig. 6(a) presents the average energy consumption. Here, it is possible to notice that smaller α values lead to higher energy consumption, while higher α leads to smaller energy values, confirming the goodness of the proposed policy. Interestingly, selecting $\alpha = 0$ can lead to almost doubled energy consumption over $\alpha = 1$. Meanwhile, concerning β, it is possible to notice a slight increment of consumed energy for higher β values. This behaviour is expected, as higher β values force Magister to promote devices depending mostly on their models' reached accuracy, discarding their energy efficiency.

Fig. 6(b) presents the average time execution. Here, it is possible to notice that smaller α values lead to smaller latency, while higher α leads to time increments. Meanwhile, β's impact is less evident, probably due to the fact that selecting devices with high accuracy improvements usually leads to improved convergence time. Differently from the energy analysis, the difference in execution time is confined as it is possible to save at most only a couple of hours when selecting properly α and β.

Fig. 6(c) presents the average number of rounds required to converge to 97% of accuracy. Here, it is possible to notice that α and β values do not impact clearly the convergence time. Indeed, convergence time is mostly influenced by the quality of the data that each device holds rather than its efficiency.

Given the results of Fig. 6, it is possible to select the best α and β for EneA-FL. Throughout our experiments, we select $\alpha = 0.6$ and $\beta = 40$ to be the best EneA-FL setup, as it allows to consume only 1.3 MJ of energy, while requiring 12.7 h to complete 30 federation rounds and converge in 10.8 steps. Finally, these results show that Magister can define hybrid systems that adapt to the scenario at hand depending on the chosen α and β values. Indeed, in a federation setup where energy represents the most relevant component we suggest selecting high α and small β values. Meanwhile, when the latency is the most valuable aspect we suggest selecting small α. Finally, when both energy and latency are relevant, Magister can reach improved efficiency by selecting a middle ground between the previous two options.

Having obtained the best α and β setup, we compare our proposed solution with the standard FL setup – which relies on random node selection – and the OORT [46] and its extended version (OORTv2) [47] in Table 3. For a fair comparison, we compare both the best and worst Magister setups. The best setup is achieved for $\alpha = 0.6$ and $\beta = 40$, while the worst counterpart is obtained by selecting $\alpha = 0$ and $\beta = 100$. The results highlight that Magister offers a favourable solution for minimising energy consumption and execution time against all baselines.

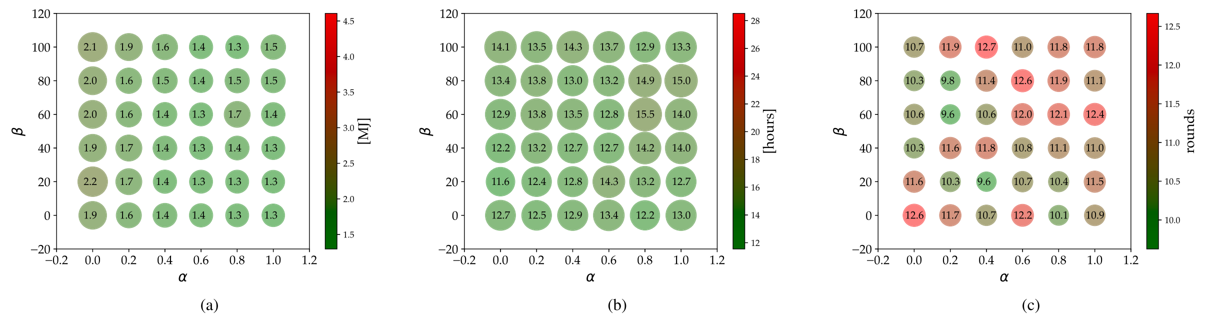


Fig. 6. Impact of (α, β) values on energy consumption (a), time execution (b) and rounds required to converge (c) on MNIST dataset.

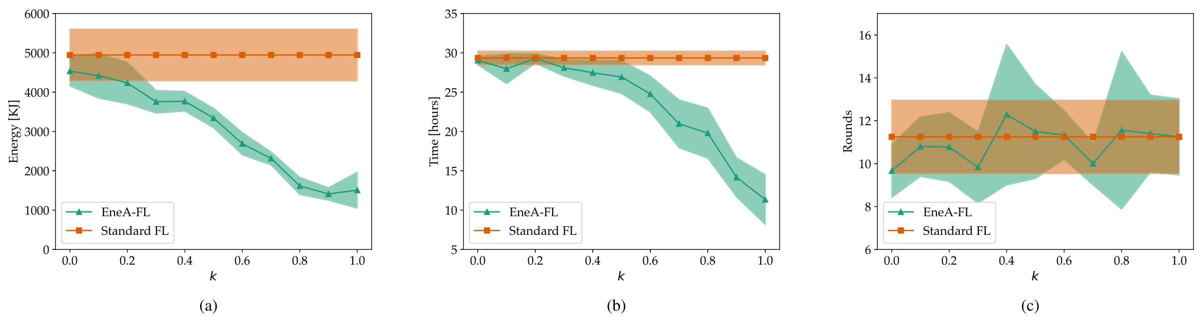


Fig. 7. Impact of k values on energy consumption (a), time execution (b) and rounds required to converge (c) on MNIST dataset.

Meanwhile, Magister does not impact the convergence time—i.e., the number of rounds required to reach 97% of accuracy. The best Magister setup consumes 3.5 times less energy than its standard FL counterpart and 30% less energy than OORT which represents the best baseline. These findings highlight the significant advantages of the Magister approach, offering substantial energy savings and improved training efficiency when compared to standard FL, making it a highly promising and effective solution for FL scenarios.

6.2.2. k values

As expressed in Eq. (7), Magister relies on a hybrid worker selection strategy, where a portion of workers is selected from the best workers of previous rounds and the remaining portion is picked randomly from the devices available. The random component is used to allow Magister to explore the effectiveness of federation devices throughout the federation history and avoid getting stuck in selecting always the same devices. The balance between the number of workers selected via the effectiveness metric and the number of randomly selected workers is given by the parameter k in Eq. (7). To assess the relevance of k we compare EneA-FL against a standard FL setup, while varying k, aiming at identifying the best k for deploying EneA-FL.

Fig. 7 shows the results for the ablation study on the values of k. We let k vary between 0 and 1. Here, $k = 0$ represents the setup where EneA-FL and random node selection are equivalent, as all devices are selected randomly also in EneA-FL. Meanwhile, $k = 1$ represents the EneA-FL setup where workers are only selected based on their energy effectiveness metric and no worker is selected randomly. Figs. 7(a)–7(c) highlight that higher values of k allow the federation to reach smaller energy consumption and execution time. Meanwhile, a similar convergence time is achieved for all k setups. Interestingly, the results show a high level of energy and time savings even for small values of k—e.g., $k = 0.5$. As a result, Magister shows that it is possible to save a high amount of energy and time even when a large part of the workers are selected randomly. From the obtained results, it is possible to identify the best k value, that we select to be equal to 0.8. We avoid considering $k = 1$ as we want to allow Magister to explore all federation devices over its history. In its best setup, Magister reaches 3 times less energy consumption, while requiring almost 2 times less time to complete the 30 federation rounds.

Table 4
Average accuracy on each dataset for EneA-FL against selected baselines when the federation is optimised given an energy budget. For each dataset, we highlight in **green** the best approach.

Dataset	Budget	Standard FL	OORT [46]	OORTv2 [47]	EneA-FL
MNIST	1 MJ	95.3%	95.4%	95.7%	97.4%
Sent140	500 kJ	72.1%	72.3%	72.3%	72.5%
NBIoT	50 kJ	87.8%	87.5%	86.7%	89.3%

6.3. Resources budget

In real-world scenarios, one desideratum of distributed optimisation scenarios such as FL is to identify efficient processes over limited resources budgets. In particular, we here consider scenarios where the federation network has a pre-defined budget of energy or time that can be invested into the FL optimisation. Indeed, when considering FL setups in the real world, it is common to impose an upper bound on the time that the FL can take for optimising the model at hand. Similar requirements can be expressed for the total amount of energy that the FL process should take to reduce costs and have a restricted environmental impact. Therefore, in order to assess the performance of EneA-FL over limited resource budgets, we compare its performance – i.e., accuracy – against the available baselines when we set energy and time limits (budgets) over the MNIST, Sent140 and N-BaIoT datasets.

Table 4 shows the average performance – over 10 iterations – achieved by the federation when training using the given energy budget. The EneA-FL policy vastly outperforms the selected baselines, achieving higher accuracy over all datasets and showing a statistically significant improvement over the experiment iterations. This is due to the higher longevity achieved by the federation when implementing the energy management policy—i.e., Magister. Indeed, the number of optimisation rounds that the federation can survive with the given energy budget is on average more than doubled with respect to the standard FL policy. This behaviour can be seen in Fig. 8, where we plot the average selected devices distribution over federation rounds on the MNIST training. On the x axis it is possible to notice that the baselines

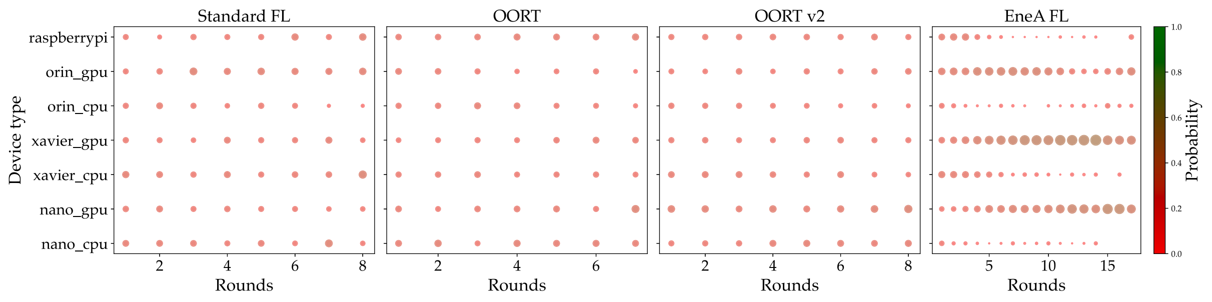


Fig. 8. Selected devices distribution over rounds for various client selection policies when federation is optimised given an energy budget of 1 MJ on the MNIST dataset.

Table 5
Average accuracy on each dataset for EneA-FL against selected baselines when the federation is optimised given a time budget. For each dataset, we highlight in **green** the best approach.

Dataset	Budget	Standard FL	OORT [46]	OORTv2 [47]	EneA-FL
MNIST	8 h	96.4%	96.2%	96.1%	97.0%
Sent140	4 h	72.3%	72.5%	72.4%	72.9%
NBIoT	10 m	87.9%	88.0%	87.8%	89.9%

survive at most 8 federation rounds, while EneA-FL reaches up to 17 optimisation rounds.

To assess the effectiveness of our policy in selecting the most efficient devices, we study the effect of the Magister policy on the device selection distribution. We plot the average selected devices distribution over the optimisation rounds in Fig. 8—one box for each client selection policy. The size and the colour of each blob represent the probability of each device type being selected for a specific round of the federation. Interestingly, all baselines behave similarly to the random context, where every device has the same uniformly distributed probability of being selected. Meanwhile, for EneA-FL the probability of selecting the Jetson Nano with GPU and the Jetson Xavier with GPU increases over the number of rounds, reaching the point where these devices represent almost the totality of devices selected for the last time stages of the federation. Similarly, the probability of selecting very slow and energy-hungry devices such as the Raspberry Pi v4 decreases over time, enabling efficiency improvements of FL. This behaviour highlights the quality of our policy for selecting efficient devices over their counterparts.

Table 5 shows the performance improvements obtained by EneA-FL against the available worker selection approaches when time budgets are considered.

6.4. Number of clients

Given the Magister focus on client selection, we here analyse the impact of the number of selected devices per round on Magister effectiveness. The number of workers selected at each federation step represents a fundamental parameter of FL settings, as it defines the broadness of data gathered at each optimisation step. Selecting a higher amount of workers for each round leads to a higher number of updates for each round, but also to increased energy and latency. Therefore, identifying smart workers selection policies such as Magister represents a key aspect to efficiently optimise the federation while avoiding selecting a high number of clients for each round.

To study the impact of the number of clients per round on the federation, we consider selecting n workers per round and let n vary between 10 and 80. Overall the federation consists of 100 devices, therefore n ranges from selecting only a small portion of the federation workers to almost selecting them all. We perform 10 federation optimisation experiments for each n, letting the federation converge to 97% accuracy when the optimisation process is stopped. Training is performed over the MNIST dataset and for each experiment we keep track of the (i)overall energy consumption; and (ii)total execution time.

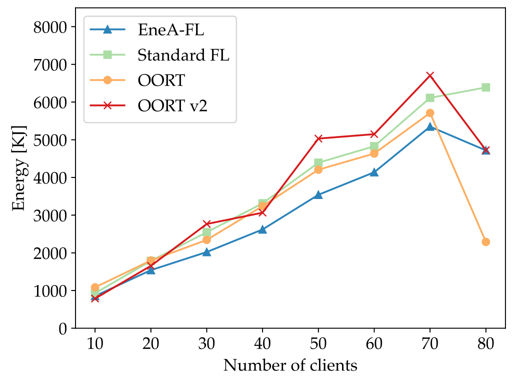
Fig. 9 shows the results of our experiments. More in detail, Fig. 9(a) presents the results concerning energy consumption. Here, it is relevant to notice how Magister outperforms all selection baselines for almost every value of n. For small n values – e.g. $n = 30$, $n = 40$ – EneA-FL requires almost 30% less amount of energy to converge to the same accuracy level. Meanwhile, for higher ns the difference is less evident. As the number of workers selected increases, Magister ends up selecting both efficient and inefficient devices – since the number of efficient devices is limited –, therefore decreasing the advantage of smart device selection. Concerning the total execution time (Fig. 9(b)), EneA-FL seems to perform similarly to the available client selection baselines, while outperforming the standard FL setup.

6.5. Device discharge

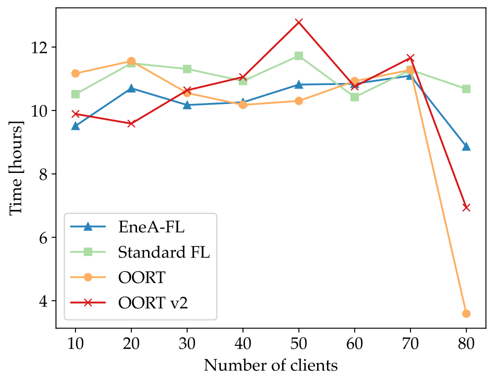
One of the issues to face when dealing with FL scenarios in edge and IoT domains is linked with the lifetime span of edge devices. In particular, in real-world setups, it is common for the edge devices to be battery-powered. Therefore, the devices belonging to the federation could suffer discharging issues, leaving the optimisation process unattended. To study if – and to what extent – the proposed energy management policy can help in these scenarios, we here study the effect of device discharging on EneA-FL and compare it to the standard FL scenario.

We model the discharging process of devices as an exponentially distributed event over the federation rounds that a device can complete and set its average value to be equal to a random value between 1 and 5. Therefore, in this experimental setup, each device belonging to the federation is capable of completing a variable number of federation rounds, after which it discharges completely and stops sending updates if selected. To account for the discharge process, the proposed energy management mechanism relies on a handicap h value that is assigned to the energy effectiveness score of each worker whenever it does not provide an update to the aggregator (recall from Section 4.2.3). In our experiments, we set $h = 5$ and measure the average number of discharged selected workers for each round. The measurements are obtained over ten iterations of the experiments to account for process variability.

Fig. 10 shows the percentage of dead selected devices over the federation round, comparing EneA-FL with the selected baselines. As expected, at first the number of discharged devices is zero, as it is considered impossible for devices to join the federation when already dead. The number of selected devices that cannot produce updates increases over the first few steps of the federation process for both approaches, as it is at this point that a few devices start to drain their batteries. However, after a few rounds, the behaviours of EneA-FL and the baselines diverge. Indeed, all baselines keep selecting devices unable to produce updates, reaching peaks of up to 80% of selected workers with drained batteries. Meanwhile, the energy management approach kicks in for EneA-FL, showing a stable percentage of dead



(a)



(b)

Fig. 9. Impact of the number of selected clients per round on energy consumption (a), and time execution (b) on MNIST dataset. Here, we consider training the federation to reach a target accuracy of 97%.

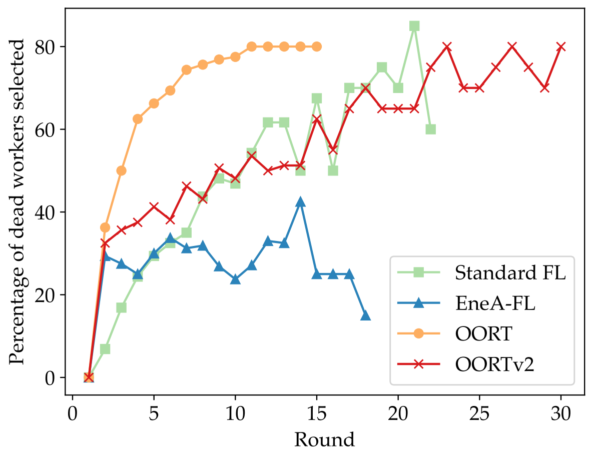


Fig. 10. Percentage of selected dead devices for each round of optimisation.

devices selection. EneA-FL allows for this percentage to stay always below 40%. This is thanks to the handicap given to devices that do not produce any update. Moreover, it is also possible to notice that the energy management mechanism allows the federation to converge more rapidly, as the updates are more consistently produced by workers. Finally, it is relevant to notice that although EneA-FL assigns handicap to inactive devices, it is still impossible to avoid completely selecting drained devices. Indeed, part of the EneA-FL selection process is random. Moreover, it is also favourable to avoid disregarding completely the drained devices, as they could become available once again if charged during the federation optimisation process.

6.6. *Impact of device type heterogeneity*

When considering real-world scenarios where FL frameworks can be deployed, one commonly changing aspect is represented by the heterogeneity of device types belonging to the federation, along with their distribution. In this context, we refer to device type as to their hardware component, thus indicating the seven different devices selected for our experimentation. The distribution of these devices may vary radically depending on the scenario at hand and this variability can impact greatly the performance of the federation process. Therefore, it is required to analyse the flexibility of a given FL framework towards the different setups of device distributions that can be encountered. Therefore, we here consider investigating how the device distribution can impact the effectiveness of the proposed EneA-FL framework.

To study the impact of devices distribution on EneA-FL we consider running different experiments where the distribution of device types differ vastly. More in detail, we focus on the devices that are found to be more efficient (see Fig. 5) and define three experimental setups where the probability of the federation containing those devices varies. In particular, we focus on the Jetson Nano with GPU and Jetson Xavier with GPU devices. We then define three experimental setups, namely:

- *Likely setup*, where Jetson Nano with GPU and Jetson Xavier with GPU are more likely to appear in the federation. Here, the probability of a random device being one of these two types is equal to 0.25.
- *Uniform setup*, where Jetson Nano with GPU and Jetson Xavier with GPU are equally likely to appear in the federation. Here, the probability of a random device being one of these two types is 0.14, similarly to the uniform device type distribution scenario considered so far.
- *Rare setup*, where Jetson Nano with GPU and Jetson Xavier with GPU are less likely to appear in the federation. Here, the probability of a random device being one of these two types is equal to 0.05.

Finally, we study whether EneA-FL and the standard FL selection policy can select efficient devices. We focus solely on the standard FL baseline, as previous experiments highlight the similarity between available client selection policies and standard FL in terms of the distribution of selected devices—see Fig. 8.

Fig. 11 shows the distribution of the selected device types over the federation rounds for the standard FL scenario. Similarly to Fig. 8, the size and the colour of each blob represent the probability of each device type being selected for a specific round of the federation. As expected, for the random selection policy, the distribution of the selected devices follows the distribution of their deployment. Here, the probability of an efficient device to be selected is equal throughout the optimisation, being higher for the *likely setup* (Fig. 11(a)) and smaller for the *rare setup* (Fig. 11(c)).

Fig. 12 shows the same distribution study for the EneA-FL setup. Here, it is possible to notice that the probability of efficient devices being selected increases over time, independently of their likelihood of being in the federation or not. Indeed, in the *likely setup* (Fig. 12(a)) the Jetson Nano with GPU and Jetson Xavier with GPU end up being almost the only devices selected from the federation mechanism. Even in the *rare setup* (Fig. 12(c)) the Jetson Nano with GPU and Jetson Xavier with

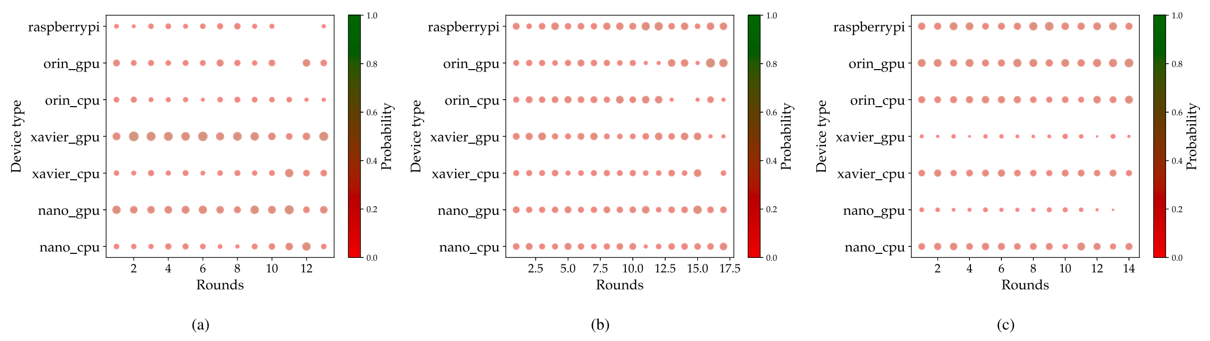


Fig. 11. Distribution of selected devices for the standard FL policy over rounds when more efficient devices (Jetson Nano with GPU and Jetson Xavier with GPU) are more likely (a), equally likely (b) or least likely to be selected.

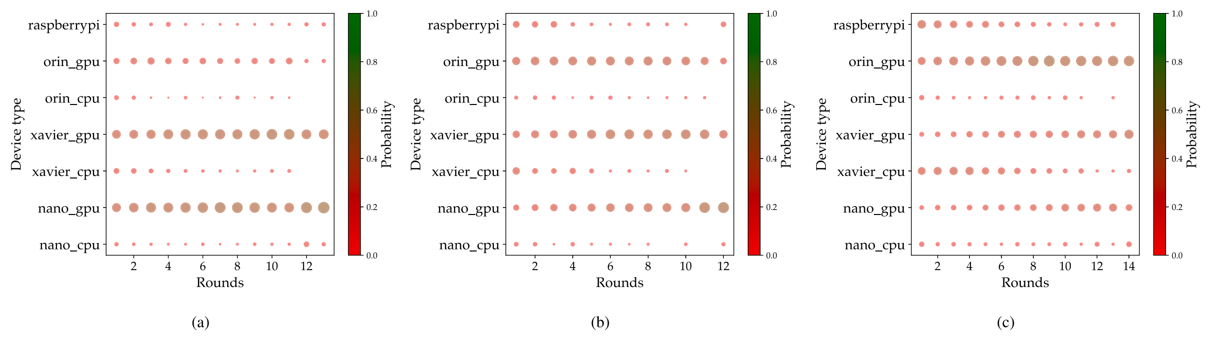


Fig. 12. Distribution of selected devices for the EneA-FL policy over rounds when more efficient devices (Jetson Nano with GPU and Jetson Xavier with GPU) are more likely (a), equally likely (b) or least likely (c) to be selected.

GPU end up being frequently selected devices, although there are only a handful of those devices in the federation. Moreover, in the *rare setup* it is relevant to notice that the other type of device popularly selected ends up being the Jetson Orin with GPU which represents the third most efficient solution according to our measurements. Finally, in all setups, it is possible to see that the least efficient devices – e.g., Jetson Orin without GPU and Jetson Nano without GPU – are also the least popular selections, independently of their proportion.

The results show the superiority of EneA-FL over the standard FL setups in terms of flexibility against device type distribution. Indeed, EneA-FL is capable of selecting efficient devices even when such devices represent a very small component of the federation devices.

7. Conclusion and future work

In this paper, we introduce EneA-FL, a novel serverless computing framework for FL in fog and constrained environments. In these environments, the resource management of devices participating in the learning process represents a fundamental component to take into account, as possible discharging issues and high-update latency can hinder the overall optimisation process. To tackle this issue, EneA-FL presents a promising solution to enhance worker selection in FL applications focusing on energy awareness, latency reduction and performance improvements. To achieve its goal EneA-FL relies on three novel components, namely (i) *Energon* – a novel energy monitoring tool –; (ii) *Furcifer* – an ad-hoc orchestrator –; and (iii) *Magister*—an hybrid energy management process. EneA-FL's unique hybrid composition and energy-conscious approach relies on a reputation system to balance energy, latency, and performance of each device, by selecting the most appropriate for each learning step. Extensive experiments show that EneA-FL yields remarkable results, demonstrating a 30% to 60% reduction in energy consumption, and faster convergence of models to the target accuracy when compared to available client selection policies.

In the future, we plan to expand the evaluation of EneA-FL to encompass diverse learning tasks beyond the ones explored in this paper. This will involve tackling challenges in graph learning [71,72] and explainable AI [73,74], thus demonstrating the versatility and applicability of our framework across various domains and cutting-edge research areas.

CRediT authorship contribution statement

Andrea Agiollo: Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Paolo Bellavista:** Supervision, Writing – review & editing. **Matteo Mendula:** Conceptualization, Investigation, Software, Writing – original draft, Methodology, Writing – review & editing. **Andrea Omicini:** Supervision, Writing – review & editing.

Declaration of competing interest

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

Data availability

Links to all data and source codes are available in the paper.

Acknowledgments

This paper was partially supported by (i) the CHIST-ERA IV project "Expectation" (CHIST-ERA-19-XAI-005), co-funded by EU and the Italian MUR; (ii) the "ENGINES – ENGineering INtElligent Systems around intelligent agent technologies" project (20229ZXBZM) funded by the PRIN 2022 MUR program; (iii) the "RESTART" project (PE00000001) under the NRRP MUR program funded by the EU - NGEU; and (iv) the "JOULE" project (2022TMT4WA) funded by the PRIN 2022 MUR program.

References

[1] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, Y. Gao, A survey on federated learning, Knowl. Based Syst. 216 (2021) 106775, http://dx.doi.org/10.1016/j.knosys.2021.106775.

[2] L. Lyu, H. Yu, Q. Yang, Threats to federated learning: A survey, 2020, CoRR abs/2003.02133. URL: https://arxiv.org/abs/2003.02133.

[3] M. Aledhari, R. Razzak, R.M. Parizi, F. Saeed, Federated learning: A survey on enabling technologies, protocols, and applications, IEEE Access 8 (2020) 140699–140725, http://dx.doi.org/10.1109/ACCESS.2020.3013541.

[4] A. Agiollo, E. Bardhi, M. Conti, N. Dal Fabbro, R. Lazzeretti, Anonymous federated learning via named-data networking, Future Gener. Comput. Syst. 152 (2024) 288–303, http://dx.doi.org/10.1016/j.future.2023.11.009.

[5] A. Agiollo, A. Rafanelli, M. Magnini, G. Ciatto, A. Omicini, Symbolic knowledge injection meets intelligent agents: QoS metrics and experiments, Auton. Agents Multi-Agent Syst. 37 (2023) 27:1–27:30, http://dx.doi.org/10.1007/s10458-023-09609-6.

[6] A. Agiollo, A. Rafanelli, A. Omicini, Towards quality-of-service metrics for symbolic knowledge injection, in: WOA 2022–23rd Workshop from Objects to Agents, in: CEUR Workshop Proceedings, vol. 3261, RWTH Aachen University, Sun SITE Central Europe, 2022, pp. 30–47, URL: http://ceur-ws.org/Vol-3261/paper3.pdf.

[7] T. Zhang, L. Gao, C. He, M. Zhang, B. Krishnamachari, A.S. Avestimehr, Federated learning for the internet of things: Applications, challenges, and opportunities, IEEE Internet Things Mag. 5 (2022) 24–29, URL: http://dx.doi.org/10.1109/IOTM.004.2100182.

[8] A. Agiollo, M. Conti, P. Kaliyar, T. Lin, L. Pajola, DETONAR: Detection of routing attacks in RPL-based IoT, IEEE Trans. Netw. Serv. Manag. 18 (2021) 1178–1190, URL: https://ieeexplore.ieee.org/document/9415869.

[9] A. Agiollo, A. Omicini, Load classification: A case study for applying neural networks in hyper-constrained embedded devices, Appl. Sci. 11 (2021) URL: https://www.mdpi.com/2076-3417/11/24/11957, special Issue Artificial Intelligence and Data Engineering in Engineering Applications.

[10] J. Sevilla, L. Heim, A. Ho, T. Besiroglu, M. Hobbhahn, P. Villalobos, Compute trends across three eras of machine learning, in: International Joint Conference on Neural Networks, IJCNN 2022, Padua, Italy, July 18-23, 2022, IEEE, 2022, pp. 1–8, URL: http://dx.doi.org/10.1109/IJCNN55064.2022.9891914.

[11] O. Rudyy, M. Garcia-Gasulla, F. Mantovani, A. Santiago, R. Sirvent, M. Vázquez, Containers in HPC: A scalability and portability study in production biological simulations, in: 2019 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2019, Rio de Janeiro, Brazil, May 20-24, 2019, IEEE, 2019, pp. 567–577, URL: http://dx.doi.org/10.1109/IPDPS.2019.00066.

[12] L.P. Dewi, A. Noertjahyana, H.N. Palit, K. Yedutun, Server scalability using kubernetes, in: 2019 4th Technology Innovation Management and Engineering Science International Conference (TIMES-ICON), 2019, pp. 1–4, http://dx.doi.org/10.1109/TIMES-iCON47539.2019.9024501.

[13] Z. Zhong, M. Xu, M.A. Rodriguez, C. Xu, R. Buyya, Machine learning-based orchestration of containers: A taxonomy and future directions, ACM Comput. Surv. 54 (2022) URL: http://dx.doi.org/10.1145/3510415.

[14] M.M. Rovnyagin, A.S. Hrapov, A.V. Guminskaia, A.P. Orlov, Ml-based heterogeneous container orchestration architecture, in: 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2020, pp. 477–481, http://dx.doi.org/10.1109/EIConRus49466.2020.9039033.

[15] Z. Zhong, M. Xu, M.A. Rodriguez, C. Xu, R. Buyya, Machine learning-based orchestration of containers: A taxonomy and future directions, ACM Comput. Surv. 54 (2022) 217:1–217:35, URL: http://dx.doi.org/10.1145/3510415.

[16] P. Xu, S. Shi, X. Chu, Performance evaluation of deep learning tools in docker containers, in: 2017 3rd International Conference on Big Data Computing and Communications (BIGCOM), 2017, pp. 395–403, URL: http://dx.doi.org/10.1109/BIGCOM.2017.32.

[17] A. Grafberger, M. Chadha, A. Jindal, J. Gu, M. Gerndt, FedLess: Secure and scalable federated learning using serverless computing, in: 2021 IEEE International Conference on Big Data (Big Data), 2021, pp. 164–173, http://dx.doi.org/10.1109/BigData52589.2021.9672067.

[18] P. Singh, M. Masud, M.S. Hossain, A. Kaur, G. Muhammad, A. Ghoneim, Privacy-preserving serverless computing using federated learning for smart grids, IEEE Trans. Ind. Inform. 18 (2022) 7843–7852, http://dx.doi.org/10.1109/TII.2021.3126883.

[19] S. Yi, C. Li, Q. Li, A survey of fog computing: Concepts, applications and issues, in: Proceedings of the 2015 Workshop on Mobile Big Data, Mobidata '15, Association for Computing Machinery, 2015, pp. 37–42, URL: http://dx.doi.org/10.1145/2757384.2757397.

[20] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, O. Rana, Fog computing for the internet of things, ACM Trans. Internet Technol. 19 (2019) 1–41, URL: http://dx.doi.org/10.1145/3301443.

[21] M. Ghobaei-Arani, A. Souri, A.A. Rahmanian, Resource management approaches in fog computing: a comprehensive review, J. Grid Comput. 18 (2019) 1–42, URL: http://dx.doi.org/10.1007/s10723-019-09491-1.

[22] A.M.A. Hamdi, F.K. Hussain, O.K. Hussain, Task offloading in vehicular fog computing: State-of-the-art and open issues, Future Gener. Comput. Syst. 133 (2022) 201–212, URL: http://dx.doi.org/10.1016/j.future.2022.03.019.

[23] M. Hussein, M. Mousa, Efficient task offloading for IoT-based applications in fog computing using ant colony optimization, IEEE Access 8 (2020) 37191–37201, URL: http://dx.doi.org/10.1109/ACCESS.2020.2975741.

[24] H. Zhou, T. Wu, X. Chen, S. He, D. Guo, J. Wu, Reverse auction-based computation offloading and resource allocation in mobile cloud-edge computing, IEEE Trans. Mob. Comput. 22 (2023) 6144–6159, URL: http://dx.doi.org/10.1109/TMC.2022.3189050.

[25] F. Shan, J. Luo, J. Jin, W. Wu, Offloading delay constrained transparent computing tasks with energy-efficient transmission power scheduling in wireless iot environment, IEEE Internet Things J. 6 (2019) 4411–4422, http://dx.doi.org/10.1109/JIOT.2018.2883903.

[26] J. Weiner, N. Agarwal, D. Schatzberg, L. Yang, H. Wang, B. Sanouillet, B. Sharma, T. Heo, M. Jain, C. Tang, D. Skarlatos, TMO: transparent memory offloading in datacenters, in: ASPLOS '22: 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Lausanne, Switzerland, 28 February 2022 - 4 March 2022, ACM, 2022, pp. 609–621, URL: http://dx.doi.org/10.1145/3503222.3507731.

[27] W. Zhang, Z. Zhang, H. Chao, Cooperative fog computing for dealing with big data in the internet of vehicles: Architecture and hierarchical resource management, IEEE Commun. Mag. 55 (2017) 60–67, URL: http://dx.doi.org/10.1109/MCOM.2017.1700208.

[28] H. Gupta, A. Vahid Dastjerdi, S.K. Ghosh, R. Buyya, IFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments, Softw. - Pract. Exp. 47 (2017) 1275–1296, URL: http://dx.doi.org/10.1002/spe.2509.

[29] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F.D. Rose, R. Buyya, CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Softw. - Pract. Exp. 41 (2010) 23–50, URL: http://dx.doi.org/10.1002/spe.995.

[30] C. Puliafito, D.M. Gonçalves, M.M. Lopes, L.L. Martins, E. Madeira, E. Mingozzi, O. Rana, L.F. Bittencourt, MobFogSim: Simulation of mobility and migration for fog computing, Simul. Model. Pract. Theory 101 (2020) 102062, URL: https://www.sciencedirect.com/science/article/pii/S1569190X19301935, modeling and Simulation of Fog Computing.

[31] X. Zhao, L. Zhao, K. Liang, An energy consumption oriented offloading algorithm for fog computing, in: Quality, Reliability, Security and Robustness in Heterogeneous Networks - 12th International Conference, QShine 2016, Seoul, Korea, July 7-8, 2016, Proceedings, in: Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 199, Springer, 2016, pp. 293–301, URL: http://dx.doi.org/10.1007/978-3-319-60717-7_29.

[32] X. Meng, W. Wang, Z. Zhang, Delay-constrained hybrid computation offloading with cloud and fog computing, IEEE Access 5 (2017) 21355–21367, http://dx.doi.org/10.1109/ACCESS.2017.2748140.

[33] S. Ahn, M. Gorlatova, M. Chiang, Leveraging fog and cloud computing for efficient computational offloading, in: 2017 IEEE MIT Undergraduate Research Technology Conference (URTC), IEEE, 2017, http://dx.doi.org/10.1109/urtc.2017.8284203.

[34] D.C. Nguyen, M. Ding, P.N. Pathirana, A. Seneviratne, J. Li, H.V. Poor, Federated learning for internet of things: A comprehensive survey, IEEE Commun. Surv. Tutor. 23 (2021) 1622–1658, URL: http://dx.doi.org/10.1109/COMST.2021.3075439.

[35] A. Imteaj, U. Thakker, S. Wang, J. Li, M.H. Amini, A survey on federated learning for resource-constrained IoT devices, IEEE Internet Things J. 9 (2022) 1–24, http://dx.doi.org/10.1109/JIOT.2021.3095077.

[36] S. Wu, G. Li, F. Chen, L. Shi, Training and inference with integers in deep neural networks, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3 2018, Conference Track Proceedings, OpenReview.net, 2018, URL: https://openreview.net/forum?id#HJGXzmspb.

[37] P. Jain, A. Jain, A. Nrusimha, A. Gholami, P. Abbeel, K. Keutzer, I. Stoica, J. Gonzalez, Checkmate: Breaking the memory wall with optimal tensor rematerialization, in: Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020, mlsys.org, 2020, URL: https://proceedings.mlsys.org/paper_files/paper/2020/hash/0b816ae8f06f8dd3543dc3d9ef196cab-Abstract.html.

[38] T. Chen, B. Xu, C. Zhang, C. Guestrin, Training deep nets with sublinear memory cost, 2016, CoRR abs/1604.06174. URL: http://arxiv.org/abs/1604.06174.

[39] H. Cai, C. Gan, L. Zhu, S. Han, TinyTL: Reduce memory, not parameters for efficient on-device learning, in: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual, 2020, URL: https://proceedings.neurips.cc/paper/2020/hash/81f7acabd411274fcf65ce2070ed568a-Abstract.html.

[40] S. Trindade, L.F. Bittencourt, N.L.S. da Fonseca, Management of resource at the network edge for federated learning, 2021, CoRR abs/2107.03428. URL: https://arxiv.org/abs/2107.03428.

[41] Z. Yang, M. Chen, W. Saad, C.S. Hong, M. Shikh-Bahaei, Energy efficient federated learning over wireless communication networks, IEEE Trans. Wireless Commun. 20 (2021) 1935–1949, URL: http://dx.doi.org/10.1109/TWC.2020.3037554.

[42] C.W. Zaw, C.S. Hong, A decentralized game theoretic approach for energy-aware resource management in federated learning, in: IEEE International Conference on Big Data and Smart Computing, BigComp 2021, Jeju Island, South Korea, January 17-20, 2021, IEEE, 2021, pp. 133–136, URL: http://dx.doi.org/10.1109/BigComp51126.2021.00033.

[43] J.F. Nash Jr., Equilibrium points in n-person games, Proc. Natl. Acad. Sci. 36 (1950) 48–49.

[44] Y. Xu, Y. Liao, H. Xu, Z. Ma, L. Wang, J. Liu, Adaptive control of local updating and model compression for efficient federated learning, IEEE Trans. Mob. Comput. (2022) URL: http://dx.doi.org/10.1109/TMC.2022.3186936.

[45] Y. Cui, K. Cao, G. Cao, M. Qiu, T. Wei, Client scheduling and resource management for efficient training in heterogeneous IoT-edge federated learning, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 41 (2022) 2407–2420, URL: http://dx.doi.org/10.1109/TCAD.2021.3110743.

[46] F. Lai, X. Zhu, H.V. Madhyastha, M. Chowdhury, Oort: Efficient federated learning via guided participant selection, in: 15th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2021, July 14-16, 2021, USENIX Association, 2021, pp. 19–35, URL: https://www.usenix.org/conference/osdi21/presentation/lai.

[47] A. Arouj, A.M. Abdelmoniem, Towards energy-aware federated learning on battery-powered clients, 2022, CoRR abs/2208.04505. URL: http://dx.doi.org/10.48550/arXiv.2208.04505.

[48] Y.J. Cho, J. Wang, G. Joshi, Client selection in federated learning: Convergence analysis and power-of-choice selection strategies, 2020, CoRR abs/2010.01243. URL: https://arxiv.org/abs/2010.01243.

[49] Y.G. Kim, C. Wu, AutoFL: Enabling heterogeneity-aware energy efficient federated learning, in: MICRO '21: 54th Annual IEEE/ACM International Symposium on Microarchitecture, Virtual Event, Greece, October 18-22, 2021, ACM, 2021, pp. 183–198, URL: http://dx.doi.org/10.1145/3466752.3480129.

[50] S. Wang, T. Tuor, T. Salonidis, K.K. Leung, C. Makaya, T. He, K. Chan, Adaptive federated learning in resource constrained edge computing systems, IEEE J. Sel. Areas Commun. 37 (2019) 1205–1221, http://dx.doi.org/10.1109/JSAC.2019.2904348.

[51] C. He, E. Ceyani, K. Balasubramanian, M. Annavaram, S. Avestimehr, SpreadGNN: Serverless multi-task federated learning for graph neural networks, 2021, CoRR abs/2106.02743. URL: https://arxiv.org/abs/2106.02743.

[52] B. Rabenstein, J. Volz, Prometheus: A next-generation monitoring system (talk), in: SREcon15 Europe, USENIX Association, Dublin, 2015, URL: https://www.usenix.org/conference/srecon15europe/program/presentation/rabenstein.

[53] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim, A survey and comparison of peer-to-peer overlay network schemes, IEEE Commun. Surv. Tutor. 7 (2005) 72–93.

[54] P. Bellavista, C. Giannelli, M. Mamei, M. Mendula, M. Picone, Application-driven network-aware digital twin management in industrial edge environments, IEEE Trans. Ind. Inform. 17 (2021) 7791–7801, http://dx.doi.org/10.1109/TII.2021.3067447.

[55] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, A. Zanella, Iot: Internet of threats? A survey of practical security vulnerabilities in real iot devices, IEEE Internet Things J. 6 (2019) 8182–8201, URL: http://dx.doi.org/10.1109/JIOT.2019.2935189.

[56] H. Wu, P. Wang, Node selection toward faster convergence for federated learning on non-IID data, IEEE Trans. Netw. Sci. Eng. 9 (2022) 3099–3111, URL: http://dx.doi.org/10.1109/TNSE.2022.3146399.

[57] Y.J. Cho, J. Wang, G. Joshi, Client selection in federated learning: Convergence analysis and power-of-choice selection strategies, 2020, CoRR abs/2010.01243. URL: https://arxiv.org/abs/2010.01243.

[58] H. Wu, P. Wang, Fast-convergent federated learning with adaptive weighting, IEEE Trans. Cognit. Commun. Netw. 7 (2021) 1078–1088, http://dx.doi.org/10.1109/TCCN.2021.3084406.

[59] G. Rjoub, O.A. Wahab, J. Bentahar, A. Bataineh, Trust-driven reinforcement selection strategy for federated learning on IoT devices, Computing (2022) URL: http://dx.doi.org/10.1007/s00607-022-01078-1.

[60] C. Mazzocca, N. Romandini, M. Mendula, R. Montanari, P. Bellavista, TruFLaaS: Trustworthy federated learning as a service, IEEE Internet Things J. (2023) 1, http://dx.doi.org/10.1109/JIOT.2023.3282899.

[61] X. Shen, Z. Li, X. Chen, Node selection strategy design based on reputation mechanism for hierarchical federated learning, in: 18th International Conference on Mobility, Sensing and Networking, MSN 2022, Guangzhou, China, December (2022) 14-16, IEEE, 2022, pp. 718–722, URL: http://dx.doi.org/10.1109/MSN57253.2022.00117.

[62] M. Aloqaily, I.A. Ridhawi, M. Guizani, Energy-aware blockchain and federated learning-supported vehicular networks, IEEE Trans. Intell. Transp. Syst. 23 (2022) 22641–22652, URL: http://dx.doi.org/10.1109/TITS.2021.3103645.

[63] M. Mendula, P. Bellavista, Energy-aware edge federated learning for enhanced reliability and sustainability, in: 7th IEEE/ACM Symposium on Edge Computing, SEC 2022, Seattle, WA, USA, December 5-8, 2022, IEEE, 2022, pp. 349–354, http://dx.doi.org/10.1109/SEC54971.2022.00051.

[64] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017 20-22 April 2017, Fort Lauderdale, FL, USA, in: Proceedings of Machine Learning Research, vol. 54, PMLR, 2017, pp. 1273–1282, URL: http://proceedings.mlr.press/v54/mcmahan17a.html.

[65] K. Pillutla, S.M. Kakade, Z. Harchaoui, Robust aggregation for federated learning, IEEE Trans. Signal Process. 70 (2022) 1142–1154, URL: http://dx.doi.org/10.1109/TSP.2022.3153135.

[66] X. Ma, J. Zhang, S. Guo, W. Xu, Layer-wised model aggregation for personalized federated learning, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, la, USA, June 18-24, 2022, IEEE, 2022, pp. 10082–10091, URL: http://dx.doi.org/10.1109/CVPR52688.2022.00985.

[67] S. Caldas, P. Wu, T. Li, J. Konečný, H.B. McMahan, V. Smith, A. Talwalkar, LEAF: A benchmark for federated settings, 2018, CoRR abs/1812.01097. URL: http://arxiv.org/abs/1812.01097.

[68] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, Y. Elovici, N-baIoT - network-based detection of IoT botnet attacks using deep autoencoders, IEEE Pervasive Comput. 17 (2018) 12–22, http://dx.doi.org/10.1109/MPRV.2018.03367731.

[69] R.J. Tallarida, R.B. Murray, Area under a curve: Trapezoidal and simpson's rules, in: Manual of Pharmacologic Calculations, Springer, 1987, pp. 77–81, http://dx.doi.org/10.1007/978-1-4612-4974-0_26.

[70] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E.Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, 2019, pp. 8024–8035, URL: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

[71] A. Agiollo, A. Omicini, GNN2GNN: Graph neural networks to generate neural networks, in: Uncertainty in Artificial Intelligence, in: Proceedings of Machine Learning Research, vol. 180, ML Research Press, 2022, pp. 32–42, URL: https://proceedings.mlr.press/v180/agiollo22a.html, proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence, UAI 2022 1-5 August 2022, Eindhoven, The Netherlands.

[72] A. Agiollo, E. Bardhi, M. Conti, R. Lazzeretti, E. Losiouk, A. Omicini, GNN4ifa: Interest flooding attack detection with graph neural networks, in: 2023 IEEE 8th European Symposium on Security and Privacy (EuroS & P), IEEE Computer Society, 2023, pp. 615–630, URL: https://www.computer.org/csdl/proceedings-article/eurosp/2023/651200a615.

[73] T.T. Huong, P.B. Ta, H. Kieu, N.V. Hoang, N.X. Hoang, N.T. Hung, K.P. Tran, Federated learning-based explainable anomaly detection for industrial control systems, IEEE Access 10 (2022) 53854–53872, http://dx.doi.org/10.1109/ACCESS.2022.3173288.

[74] A. Agiollo, L.C. Siebert, P.K. Murukannaiah, A. Omicini, The quarrel of local post-hoc explainers for moral values classification in natural language processing, in: D. Calvaresi, A. Najjar, A. Omicini, R. Aydoğan, R. Carli, G. Ciatto, Y. Mualla, K. Främling (Eds.), Explainable and Transparent AI and Multi-Agent Systems, in: Lecture Notes in Computer Science, vol. 14127, Springer, 2023, pp. 97–115, http://dx.doi.org/10.1007/978-3-031-40878-6_6.

Andrea Agiollo received the Bachelor of Information Engineering degree from the University of Padua, Italy, in 2018, the first master's degree in information and communication technologies for Internet and multimedia from the University of Padua in 2020, and the second master's degree in communication engineering from National Taiwan University, Taiwan, in 2020. He is currently pursuing the Ph.D. degree in computer science and engineering with the University of Bologna, Italy, in collaboration with Electrolux Professional S.P.A. From October 2022 to February 2023, he was with Delft University of Technology in the Netherlands as a visiting Ph.D. student, and from August 2023 to November 2023 he joined Purdue University with the same position. His research interests include machine learning in resource-constrained environments, neuro-symbolic artificial intelligence, explainable artificial intelligence, and network security.

Paolo Bellavista received the Ph.D. degree in computer science engineering from the University of Bologna, Italy, in 2001. He is currently a Full Professor with the University of Bologna. His research interests include middleware for mobile computing, QoS management in the cloud continuum, infrastructures for big data processing in industrial environments, and performance optimisation in wide-scale and latency-sensitive deployment environments. He serves on the Editorial Boards of IEEE Communications Surveys and Tutorials, IEEE Transactions on Network and Service Management, IEEE Transactions on Service Computing, ACM CSUR, ACM TIOT, and PMC (Elsevier). He is the Scientific Coordinator of the H2020 IoTwins Project (https://www.iotwins.eu).

Matteo Mendula received the M.Sc. degree in software engineering at the University of Bologna, Italy, in 2020. He is currently enrolled as a Ph.D. student at the Department of Computer Science and Engineering, University of Bologna. His main research interests include Big Data processing and distributed learning on the edges of the network. In particular, his research relates to architectural aspects and Machine Learning enhanced techniques in fog computing scenarios.

Andrea Omicini is Full Professor of the Alma Mater Studiorum–Universitá di Bologna, and holds a Ph.D. in Computer & Electronic Engineering. He has published over 350 articles on multi-agent systems, intelligent systems engineering, computational logic, explainable AI, agreement technologies, self-organising systems, simulation, and pervasive systems. He is currently teaching intelligent systems engineering, multi-agent systems, and distributed systems.