Edge Computing Solutions for Distributed Machine Learning

Conference Paper · July 2022

DOI: 10.1109/DASC/PiCom/CBDCom/Cy55231.2022.9927824

Fabrizio Marozzo
Università della Calabria
109 PUBLICATIONS **1,377** CITATIONS
SEE PROFILE

Alessio Orsino
Università della Calabria
9 PUBLICATIONS **26** CITATIONS
SEE PROFILE

Domenico Talia
Università della Calabria
449 PUBLICATIONS **7,290** CITATIONS
SEE PROFILE

Paolo Trunfio
Università della Calabria
160 PUBLICATIONS **3,120** CITATIONS
SEE PROFILE

Edge Computing Solutions for Distributed Machine Learning

Fabrizio Marozzo, Alessio Orsino, Domenico Talia, Paolo Trunfio
DIMES, University of Calabria, Italy
{fmarozzo, aorsino, talia, trunfio}@dimes.unical.it

***Abstract*—In recent years machine learning (ML) has achieved great results in providing solutions for many tasks such as speech recognition, sentiment analysis, email spam filters, fraud prevention and so on. The rapid spread of the Internet of Things (IoT), with billions of connected devices, has generated huge amounts of data and asks for decentralized solutions for machine learning. However, performing complex learning tasks at the edge of the network is posing great challenges in terms of efficient management of data storing, transfer, and analysis. For these reasons, a lot of research and development effort is devoted to adapt different machine learning algorithms (e.g., neural networks, ensemble algorithms, SVM, k-means) so that cooperative training and inference on local data occur directly at the edge of the network (i.e., close to where the data is generated). This scenario represents a major challenge today due to the limited capacities of edge devices, the different technologies with which these devices work and communicate, and the lack of common software stacks to easily manage them. In this paper, we analyze distributed machine learning algorithms and how they should be adapted to run at the network edge and, if needed, cooperate with the cloud to ensure low latency, energy savings, privacy preserving and scalability. In particular, we briefly discuss how the main machine learning algorithms have been adapted to work in traditional distributed platforms (such as clusters, clouds, and HPC systems) and the main research work that has led these algorithms to run on resource-constrained edge devices. Then, a layered approach is introduced and discussed for adapting machine learning algorithms on edge-cloud architectures. This is done by taking into account the application and device constraints and the features of the multi-layer supporting architecture. Finally, we conclude the paper by describing some application scenarios that can benefit from this approach.**

***Index Terms*—Machine learning, distributed machine learning, Internet of Things, edge computing, cloud computing, edge-cloud continuum**

I. Introduction

In the last decades artificial intelligence, especially machine and deep learning, have become established as a solution to solve several tasks that are part of our daily life, such as speech recognition, email spam filters, fraud prevention and others. One of the enabling factors for the development of machine learning solutions in recent times has been Big Data [1]. In fact, the availability of huge amounts of data through which to train learning algorithms, and the growing computational capacities available today provided a significant boost to machine learning, which is able to extract potentially useful information from data for decision making. Traditional approaches to machine learning rely on the storage and processing of such data in conventional distributed systems, which for years have proved to be the ideal solution for solving complex learning tasks, especially for those applications that do not have low latency requirements.

However, the spread of Internet of Things (IoT) devices (i.e., Internet-connected objects that can collect and transfer data without human intervention, such as smart cameras and vehicles, wearables, and smartphones) has led to an even greater generation of data at the network edge. The network edge is generally defined as the place where a device connects to the Internet. Transferring all this data from the sources to a centralized server for collecting, processing and analyzing it through machine learning algorithms involves high communication costs, with possible repercussions on latency, which may be critical for low latency applications, such as health monitoring and security. To solve this problem, it is natural to consider processing the data as close as possible to where it is generated, i.e., the network edge where devices reside. To meet this demand, edge computing [2] has emerged as a paradigm that pushes computing tasks to the network edge.

Nonetheless, IoT devices at the edge of the network have limited computational and energy power, storage capacity and bandwidth, which makes it infeasible to fully perform heavy learning tasks on such devices. They are usually defined as resource-constrained devices, which means they can not be integrated with additional resources [3]. For these reasons, efforts in the last years have been devoted to adapting machine learning algorithms to run cooperative training and inference on local data directly on edge devices. However, this task represents a major challenge today due to the limited capacities of edge devices, the heterogeneous hardware and technologies with which these devices work and communicate, and the lack of common software stacks to easily manage them. Beyond that, security and privacy are additional critical concerns that need to be taken into account when data must be transferred to other devices or to a remote server to train a model in parallel. Communication is another challenge, as the bandwidth between edge devices can be much slower than local computation time, making it necessary to develop communication-efficient methods for the training process.

In this work, we analyze distributed machine learning algorithms and how they should be adapted to be deployed on the edge and, if needed, to cooperate with the cloud to ensure low latency, energy savings, privacy-preserving and scalability. Specifically, we briefly discuss how the main machine learning algorithms have been adapted to work in traditional distributed platforms such as clusters, clouds, and High Performance Computing (HPC) systems, and outline the state-of-the-art techniques to run these algorithms on resource-constrained edge devices. In this respect, most of the work in the literature has been devoted to deep learning applications, thus traditional machine learning techniques such as Support Vector Machine (SVM), k-means, ensemble learning (e.g., Random Forest) and so on, which, on the other hand, are frequently used in artificial intelligence applications on edge computing environments have not sufficiently investigated. Then, we discuss a solution for adapting machine learning algorithms on edge-cloud architectures, in what is called the Edge-Cloud Continuum. The Edge-Cloud Continuum leverages all the resources from the edge of the network (e.g., IoT devices) to the core (e.g., cloud data centers). Data is generated in the edge layer, where it can be first processed locally, while aggregation and partial processing are done in the intermediate nodes. Only if necessary, data is transferred to the cloud for further analysis [4].

The structure of the paper is as follows. Section II analyzes research work in the field of distributed machine learning at the Edge-Cloud Continuum. Section III presents a review of machine learning algorithms in traditional distributed high performance platforms. Section IV presents a review of machine learning algorithms proposed to be performed on resource-constrained devices at the edge of the network, both in training and inference. A brief discussion of privacy and security issues is introduced with the federated learning paradigm in Section V. Section VI introduces and discusses a layered approach for adapting machine learning algorithms on edge-cloud architectures. Section VII describes some application scenarios that can benefit from this approach and finally Section VIII concludes the paper.

II. Related work

Intelligent applications and services based on machine learning have been increasingly used in edge computing environments, mainly due to the demand for low latency in real-time scenarios. However, the resource-constrained nature of edge devices requires that they must collaborate to perform distributed training and inference and possibly be supported by cloud resources. This scenario provided the main motivation for this work. Previous work on distributed machine learning at the Edge-Cloud Continuum does not cover traditional machine learning algorithms, which instead are widely used. For example, Zhou et al. [5] summarize the main solutions and enabling technologies for model training and inference of deep neural networks on edge devices, both from an application and software perspective. Likewise, [6] reviewed emerging techniques to speed up deep learning models on edge devices.

In a similar way to us, Murshed et al. [7] conducted an analysis on both traditional machine learning and deep learning in resource-constrained edge computing environments. Particularly, they discuss common software and hardware solutions used in deep learning at the edge, but the discussion for traditional machine learning is limited to a detailed description of applications in edge computing scenarios. Another interesting work is proposed by Imteaj et al. in [3], where the federated learning paradigm is examined and the main issues in training distributed machine learning models for resource-constrained IoT devices are discussed. In [4], Rosendo et al. focus on distributed intelligence on the Edge-Cloud Continuum, by reviewing work in the fields of machine and deep learning and data analytics applied on Edge, Cloud, and Edge-Cloud architectures. However, [4] mainly focuses on frameworks and libraries for Big Data processing across the Edge-Cloud Continuum and the leading state-of-the-art simulation and deployment systems that support experimental research on the Edge, Cloud, and Edge-Cloud Continuum.

In the following sections we will discuss $i)$ how machine learning algorithms have been adapted to traditional distributed high performance platforms such as cluster, cloud and HPC systems, $ii)$ the main efforts for performing machine learning on resource-constrained devices at the edge of the network and $iii)$ privacy and security issues in distributed learning with the Federated learning paradigm. Table I summarizes the main shortcomings of the state-of-the-art in distributed machine learning at the Edge-Cloud Continuum.

III. Distributed machine learning

Generally, machine learning tasks can be classified into supervised, unsupervised, and reinforcement learning [38]. Briefly, training data is labelled in supervised learning, in contrast to unsupervised learning which does not require any label. Differently, reinforcement learning is concerned with learning from feedback coming from external interactions. Machine learning algorithms are designed to run on powerful machines, which are often equipped with acceleration hardware such as GPUs and FPGA. However, nowadays due to the growing size of training data and machine learning models, learning on single machines can not be done either efficiently or effectively due to limited hardware [39]. Distributed computing can therefore help alleviate these problems. In distributed machine learning multiple workers cooperate and communicate with each other for training a model in parallel. In particular it can be done with two different approaches: distributing the data or distributing the model [40]. In the first approach data is partitioned on the worker nodes of the distributed system, which all execute the same algorithm on different partitions. The models obtained by training the algorithm on the various partitions must then be aggregated. In the second approach, instead, the same data is processed by the worker nodes by executing different partitions of the model and the final model is therefore generated by the aggregation of all parts. This approach can be applied to all those machine learning algorithms in which parameters can be partitioned (e.g., neural networks). Another approach is based on ensemble learning, in which several instances of the same model are trained and outputs are aggregated. In all of these approaches, worker nodes can be organized in either a centralized architecture

Category	ML algorithms	Cloud	Edge	Edge-Cloud Continuum	Goal	References
Distributed machine learning (Section III)	Main machine learning algorithms (e.g., k-means, DBSCAN, SVM, Random Forest and others)	✓			Adapting ML algorithms to traditional distributed high performance systems	[8]–[23]
Machine learning on resource-constrained edge devices (Section IV)	Main machine learning algorithms (e.g., k-means, kNN, trees and others)		✓		Adapting ML algorithms to resource-constrained devices at the network edge	[24]–[30]
Federated machine learning (Section V)	Mainly gradient-descent and Random Forest			✓	Privacy and security issues in distributed learning	[31]–[37]

TABLE I: State-of-the-art in distributed machine learning at the Edge-Cloud Continuum.

(also known as parameter server) or in a decentralized one. The parameter server architecture consists of one or more servers and several workers, and the learning process is performed in an iterative manner by updating and synchronizing model parameters with central servers. In the decentralized setting, instead, each worker node communicates with its neighbors and the model is aggregated without a central node. In all approaches and architectures, the main benefit of distributed learning is to avoid the need to collect large volumes of data on a single machine to be processed, saving time and energy and increasing reliability [38].

In the following we will analyze some implementations of distributed machine learning algorithms in supervised and unsupervised settings and some distributed machine learning frameworks. It is worth noticing that all the papers analyzed proposed how to speed up traditional machine learning algorithms (k-means, DBSCAN, SVM, and so on) in traditional distributed high performance infrastructures (cluster, multi-processor and multi-node environments, and HPC platforms). Many of them are based on Big Data analysis paradigms and frameworks, such as Apache Hadoop and Spark. Therefore, they cannot be directly adapted to be deployed on edge devices, where in addition to the scalability we must take into account other concerns such as limited computing and storage capacities, energy savings, data privacy and limited bandwidth for communication. Particularly, communication overhead is one of the major challenges in edge computing environments.

A. *Distributed supervised learning*

In supervised learning, most efforts have been devoted to developing distributed classification algorithms, especially the SVM and tree-based algorithms like Random Forest. For example, [8] and [9] present a MapReduce-based distributed SVM algorithm that partitions the training data and optimizes the partitioned subsets over cloud and clusters of computers, thus reducing the training time while maintaining a good level of accuracy. In reference [10] a distributed SVM algorithm in a master–worker setting is presented. The distributed SVM is treated as a regularized optimization problem and modeled as a series of convex optimization sub-problems that are solved using optimization techniques. Dass et al. [11] proposed a distributed, scalable and communication-efficient algorithm for SVM training that uses a compact representation of the kernel matrix to reduce both computation and storage during the training process. As for the Random Forest algorithm, in [12] a parallel version for Big Data classification based on Apache Spark is presented. The algorithm is optimized using a hybrid approach that combines data and task parallelism.

B. *Distributed unsupervised learning*

In unsupervised learning, clustering is one of the most used tasks and k-means and DBSCAN (Density-based spatial clustering of applications with noise) are two of the most popular clustering algorithms. A lot of research and development efforts have been oriented to improve their performance by parallelizing and/or distributing them. For example, in [13] Zhang et al. proposed a parallel strategy for the k-means algorithm based on a data parallel distribution approach and a parameter-server architecture with dynamic load balance. Similarly, in [14] Zhao et al. proposed a parallel k-means clustering algorithm based on the MapReduce programming paradigm. More recently, in [15] the problem of distributed clustering is analyzed by Balcan et al., where the data is partitioned across nodes whose communication is restricted to the edges of a graph structure. A distributed k-means algorithm with low communication cost is described, based on the construction of a small set of points which act as a proxy for the entire data set. In [16] Qin et al. developed a distributed k-means algorithm for wireless sensor networks where each node is equipped with sensors. The proposed distributed implementation is capable of partitioning the data into groups having small in-group and large out-group distances.

In [17] Patwary et al. presented a parallel DBSCAN algorithm using graph algorithmic concepts, for shared and for distributed memory systems. Specifically, they exploit the disjoint-set data structure to break the intrinsic sequentiality of DBSCAN and use a tree-based approach to build the clusters, ensuring workload balancing. Similarly, Götz et al. [18] presented a parallel approach for DBSCAN that employs three techniques in order to break the intrinsic sequentiality of the algorithm and enhance workload balancing in distributed processing environments. These techniques are: i) a computation split heuristic for domain decomposition; ii) a data index preprocessing step, and iii) a rule-based cluster merging scheme. In [19] Chen et al. presented a parallel version of the DBSCAN algorithm in distributed environments that works by partitioning the data, then each node builds clusters independently and the sub-results will be aggregated into one final result. A very similar solution (i.e., data partitioning, parallel cluster building, and final merge) is presented in [20], where Luo et al. leverage the distributed Spark framework.

For most machine learning algorithms (even the less used ones) there is at least one implementation dedicated to distributed environments. For example, in [21] Pizzuti and Talia proposed a parallel implementation for distributed memory multicomputers of AutoClass, a clustering algorithm based on Bayesian classification.

C. Distributed machine learning frameworks

Other than specific algorithms, different parallel/distributed software frameworks include distributed machine learning libraries. For example, Apache Mahout [22] is an open source library to develop scalable machine learning algorithms. It is built on top of Hadoop and includes recommendation mining, clustering, classification, and frequent itemset mining algorithms. MLlib [23], on the other hand, is Apache Spark's machine learning library, which provides advanced data analytics with parallel machine learning algorithms such as classification, regression, clustering, and collaborative filtering built on top of Spark [41].

IV. Machine learning on resource-constrained edge devices

Deploying machine learning applications at the edge is a key opportunity for different real-world application scenarios, which can benefit from the low latency deriving from performing training and inference near to the data sources (see Section VII). However, the nature of IoT edge devices (i.e., limited computational and energy power, heterogeneity in hardware and technologies, security and communication issues) poses a great challenge in performing heavy learning tasks on such devices. This is a relatively new research field where a few systems have been proposed. Here we consider learning at the edge (i.e., edge learning) as including both the training (i.e., edge training) and the inference (i.e., edge inference) process.

A. Edge training

While inference is commonly performed on edge devices, edge training is much less common [24]. The main efforts in techniques for training machine learning models in edge devices are mostly concerned with deep learning. The goal is to get lightweight deep learning models, which can be learned collaboratively on edge devices. Mainly the literature focuses on algorithms that are based on the gradient-descent technique for training. Generally, the distributed gradient-descent learning process includes local update steps where each edge device performs gradient-descent to improve the local model parameter for minimizing a loss function on its own local dataset. Then, a global aggregation step is required where model parameters obtained by different edge devices are sent to an aggregator, which is a component that usually runs on the remote cloud. After aggregation, the updated parameters are sent back to the edge devices for the next round of iteration.

Following this approach, Wang and co-authors [25] proposed a technique to train machine learning models at the edge without cooperation with cloud servers. The technique minimizes the loss function of a learning model by using only edge devices. Local gradient-descent is performed on multiple edge devices on local data. Local models are sent to another edge device (i.e., the aggregator) that computes a weighted average and sends it back to all the edge devices for the next iteration steps. The authors show the effectiveness of this technique using only three Raspberry Pi devices and a laptop computer as experimental settings, achieving performance close to the optimum on different datasets. The gradient-descent-based distributed learning has been extensively studied from a theoretical point of view in [26] too.

B. Edge inference

The inference process usually takes place at the Edge, in order to ensure low latency and privacy of local data. In this section we discuss recent work that has proposed frameworks and algorithms for enabling inference on resource-constrained edge devices and reducing prediction costs. For example, in [27] is presented a tree-based algorithm, named Bonsai, for efficient prediction on IoT devices. Bonsai is able to maintain accuracy while minimizing model size and prediction costs. This is done by developing a tree model which learns a shallow and sparse tree. The data is then projected into a low-dimensional space in which the tree is learnt. Bonsai was deployed and evaluated on the Arduino Uno board. A similar experimental approach has been carried out in [28] by Gupta and co-authors. The authors proposed ProtoNN, an algorithm based on k-Nearest Neighbor (kNN) for accurate prediction on resource-constrained devices. ProtoNN is based on three key aspects: i) learning a small number of prototypes to represent the entire training set, ii) sparse low dimensional projection of data, and iii) joint discriminative learning of the projection and prototypes. In [29] Yazici et al. tested three different algorithms (Random Forest, SVM and Multi-Layer Perceptron) on the Raspberry Pi using ten different datasets. In particular, they evaluated performance in terms of inference process time, accuracy, and power consumption. Results show that the Random Forest algorithm had the highest accuracy, while the SVM algorithm is faster in inference and more efficient in power consumption. To reduce the size of ensemble models, in [30] a pruning method for Random Forest on resource-constrained devices is proposed, to optimize costs and accuracy. In particular, the pruning problem is posed as an integer program and solved with a large-scale primal-dual algorithm.

For what concerns deep learning, the usual approach for deploying models in edge devices is to train large and accurate models on powerful machines (i.e., cloud or cluster) and then use compression techniques (i.e., low-rank approximation, knowledge distillation, pruning, parameter quantisation) to reduce the size. In this direction, the Tiny machine learning paradigm is a fast-growing area where machine learning algorithms are capable of performing data analysis on devices with extremely low power consumption. However, compressed models often result in lower accuracy [7], thus the trade-off between accuracy and costs must be further investigated.

The research work discussed demonstrates the effectiveness of techniques to reduce data transfer and size of machine learning models, thus improving the inference performance in resource-constrained edge devices. However, none of them is intended to be deployed in a real-world edge computing environment with a number of edge devices that can also collaboratively perform the inference process.

V. Federated machine learning

Most techniques for distributed machine learning manage data in a centralized way, without considering privacy and security concerns during training or inference. In particular, updating the global model in the training process depends on the information sent by edge devices, which generally have limited defense capabilities and can be compromised by potential attacks. To meet this demand, federated learning is a machine learning paradigm whose aim is to train a centralized model while data remains distributed over a large number of clients [42]. While federated learning is widely used for gradient-descent based algorithms, privacy concerns in traditional machine learning algorithms are not sufficiently investigated.

When the federated paradigm is applied to clustering, the main goal is to group local data stored on each node that are globally similar to each other. Kumar and co-authors [31] proposed to apply the Federated Averaging technique to the k-means algorithm based on the distributed version proposed by [43]. The data produced at the edge devices is never sent to a centralized node, thus ensuring privacy preservation and latency reduction. The final model is obtained by iterative model averaging. Dennis, Li and Smith [32] developed a one-shot federated clustering scheme, named k-FED, which requires only one round of communication with a central server. Each device solves a local k-means problem and then communicates its local cluster means via message passing. Using federated learning SVM, instead, in [33] is presented a privacy-preserving federated learning system for detecting Android malware. It allows mobile devices to collaboratively train a classifier without exposing sensitive information.

Some effort was devoted to applying the federated learning paradigm in ensemble techniques, especially the Random Forest algorithm, which is one of the most used machine learning algorithms in a wide range of industrial scenarios. For example, in [34] Wu et al. proposed Pivot, a solution for privacy-preserving vertical decision tree training and prediction, which ensures that no intermediate information is exposed. The proposed solution can also be extended to tree ensemble models like Random Forest and gradient boosting decision trees. Still in the field of vertical federated learning, in [35] Yao et al. proposed a federated Random Forest algorithm designed for both efficient training and inference and a distributed system to exploit parallelism in Random Forest, achieving high partition tolerance. The system involves an efficient homomorphic cryptosystem to provide protection on data privacy as well. Also in [36] a model for vertical federated learning is proposed by Han et al., named Federated Gradient Boosting Forest, which simultaneously integrates the boosting and bagging by building the decision trees in parallel as a base learner. In [37] Liu and co-authors focused on a privacy-preserving learning system for Random Forest that achieves the same accuracy as the non-privacy preserving approach. In more detail, a learning system was developed to collaboratively train a model over different clients using the same user samples but different attributes without the need of exchanging raw data. A prediction process was also proposed for reducing the communication overhead among clients.

Although these proposals demonstrate that federated learning offers several advantages, especially scalability and data privacy, they do not consider the hardware features of edge devices, which are usually limited in computational and storage resources. They focus only on privacy and security aspects of distributed learning at the edge, assuming each edge device has available computational resources.

VI. Distributed machine learning at the Edge-Cloud Continuum

Executing machine learning algorithms in distributed environments requires (a) to separate the tasks that compose an algorithm, (b) to coordinate their execution on the different computing nodes in accordance with any dependencies that exist between them, and (c) to manage failures that can occur on computing nodes and communication links that can be unreliable. For this reason, the choice of an appropriate distributed algorithm to solve a given problem depends both on the characteristics of the problem and on the features and configuration of the system on which the algorithm will run, the type of communication and synchronization among processes that can be performed. All these problems of traditional distributed systems are amplified when we consider IoT systems characterized by limited computing capacities, problems of energy consumption and latency, different technologies and software stacks.

The approach we discuss here aims to adapt distributed versions of machine learning algorithms to IoT environments. As shown in Figure 1, in the Edge-Cloud Continuum we can identify four different layers [44]:

- *Device layer*. This is the layer at which IoT devices generate or collect data (*data collection*). This data can be generally stored in the device's storage system, both in a persistent or temporary way (*data storage*). Before storing or using it to perform analytics on the device, data can be filtered according to application requirements (*data filtering*). Then, this local data can be used to train a learning model (*local learning*), which at higher levels may be used in federated learning tasks.
- *Edge layer*. This layer is where gateways such as routers, base stations or micro data centers serve to bring computing closer to IoT devices. The goal of this proximity

is to gather the sensed data of IoT devices (*data aggregation*), preprocess and possibly cache it (*data filtering and caching*) and send it to the cloud (or fog) for storage purposes or to perform complex learning tasks that cannot be performed at this level or in that device. Here it is possible to aggregate local models learned from many devices (*model aggregation*).

- *Fog layer*. This is an intermediary layer that can benefit from the computation that is closer to the cloud and more powerful than that provided by the edge layer. Data can be stored and exploited for collective learning. In particular, after the local learning on private data at the device level, a collective training phase can take place at this level in which labels are assigned to shared and unlabelled data by means of a consensus-based algorithm.
- *Cloud layer*. This layer acts as the backbone of the network and provides persistence data storage, and powerful and very large processing resources [45] that are not available in the other layers. If needed, it can be leveraged for aggregating global models (*global learning*).

Not all of the levels described above are necessary (for example the fog is optional), but if reduced to two extreme levels device-cloud the architecture turns into a traditional cloud-based solution. The implementation of decentralized algorithms on this four-layers architecture must take into account the following aspects:

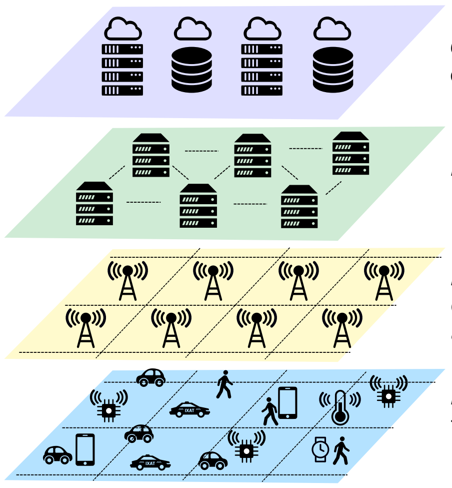
1) *Data location* - Computation is moved as close to the data as possible to minimize data movement among nodes. Since the various levels are made up of heterogeneous software components having different computing capabilities, if a task cannot be performed (or is inefficient) on a certain node of a given level, support is requested to the higher level (and so on). For example, tasks that cannot be performed on devices are offloaded to edge nodes, and so on to fog nodes up to the cloud.
2) *Geographical distribution* - It is necessary to take into account the physical distribution of the hardware components that will execute the distributed algorithm. The hardware components may be in different places far from each other. Therefore, localization of devices may influence communication overhead and algorithm performance.
3) *Algorithm features and configurations* - Each algorithm can be adapted differently than the others and there may not be a unique way of migrating algorithms (code) in these environments. There may be patterns that allow a user to define and execute classes of machine learning algorithms in IoT environments, but there are many variables to consider. Moreover, deploying distributed algorithms in hybrid cloud/edge architectures is an extremely complex problem, as there are different and abundant effective configuration parameters (NP-hard problem [46]).
4) *Task scheduling and persistence of data* - A data-aware scheduler is needed to efficiently perform the tasks that compose distributed learning algorithms. The scheduler must ensure a load balance between all the computing nodes (some nodes can have different computing power), pay more attention to highly critical tasks, and, if necessary, perform replication of the tasks to improve both execution time and fault tolerance [47]. Often a centralized or distributed master node must be identified to coordinate all these scheduling activities. Moreover, consider that temporary data may be stored on non-persistent components and therefore may be lost.
5) *Application constraints* - Functional and not functional constraints (quality of service or QoS) of the application that will use a machine learning algorithm must be respected. Often it is essential to meet crucial application and system requirements such as low latency, energy saving, network traffic reduction, privacy preservation, and high scalability. For those reasons machine learning algorithms must also be configurable in order to meet the requirements. In fact, the same algorithm can be performed differently on the architecture with different degrees of freedom (e.g., a less precise model on the edge nodes, a more accurate model on the cloud).
6) *Coding and testing* - Let consider that each layer could be implemented by different hardware and could be programmed by different software tools/libraries. This aspect makes it difficult to write and thoroughly test an algorithm. Furthermore, real tests using a large number of hardware nodes could be very expensive and inflexible, and benchmarking and setting up real experiments could be very challenging. Simulation tools result to be powerful and flexible for reproducing and testing IoT systems and networks [48]–[50].

VII. Application scenarios

Many different real-world application scenarios where distributed learning must be used, may benefit from the cooperation between edge and cloud in the Edge-Cloud Continuum can be identified. Many of them needing real-time computations, low energy consumption, high scalability and good levels of privacy can benefit from the use of edge solutions possibly integrated with cloud solutions. Here we discuss three significant scenarios where to exploit the proposed approach.

A. Smart cities

The edge computing paradigm enables vehicles and humans to connect and integrate a wide variety of services to enhance personalized experiences in smart cities [51]. This can be done using machine learning techniques able to extract knowledge from data sources at the network edge. The Edge-Cloud Continuum can offer wide and efficient support to all those advanced mobility services, such as: $i)$ moving taxis to areas of the city where it is likely to find new customers; $ii)$ providing advertisements to car drivers based on location and preferences; $ii)$ suggesting places to visit to pedestrians, based on previous places visited. The intelligence can be embedded into edge servers and devices that can collaboratively train



Cloud layer - Persistent storage, global learning
Fog layer - Data storage, meta learning/collective learning
Edge layer - Data aggregation, caching and filtering, model aggregation
Device layer - Data collection, filtering and storage, local learning

Fig. 1: The Edge-Cloud Continuum architecture.

machine learning models on the data they produce, ensuring scalability of the learning process and privacy protection of local data. A global model can be then obtained at the cloud or fog layers by aggregating models learned at the edge.

B. Industrial IoT

Industrial IoT allows machines to improve the performance and productivity of industrial processes while reducing waste [52]. Combining data from a network of sites can result in a more efficient control of material flow, and early detection, identification and elimination of production or supply bottlenecks, thus optimizing industrial operations. The layered architecture we propose can enable advanced IIoT applications, for example: i) executing machine learning tasks at the edge in order to reduce response time and bandwidth; ii) predictive maintenance analysis of industrial equipment to identify potential failures before they impact production; iii) advanced logistics management.

C. Smart healthcare

Many healthcare organizations have adopted cloud computing solutions due to the need to access services through standard mechanisms and to scale resources to perform heavy and intelligent tasks. Medical devices are seen as part of IoT applications directly connected to the network through clinical workflows. Nevertheless, the emerging availability and complexity of various types of medical devices, along with the large data volumes that such devices generate, can limit the current cloud-based IoT applications. In this scenario, smart healthcare solutions [53] can exploit Edge-Cloud Continuum solutions to continuously monitor and assist patients at home, obtain insights from hospital equipment for improving patient outcomes and optimize medical supplies.

VIII. Conclusion

Traditional distributed approaches to machine learning rely on the storage and processing of Big Data in large-grain distributed systems or clouds. They require transferring all data from sources to a centralized server. Collecting, processing and analyzing it involves high communication costs, which may be critical for low latency applications. The edge computing paradigm has emerged to meet this demand by processing data as close as possible to where it is generated. Nonetheless, IoT devices at the network edge have limited computational and energetic power, which makes it infeasible to fully perform heavy learning tasks on such devices. For these reasons, efforts, like the one we discuss, must be devoted to adapting machine learning algorithms to perform cooperative training and inference on local data available on edge devices.

References

[1] L. Belcastro, F. Marozzo, and D. Talia, "Programming models and systems for big data analysis," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 34, pp. 632–652, 2019.

[2] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[3] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 1–24, 2021.

[4] D. Rosendo, A. Costan, P. Valduriez, and G. Antoniu, "Distributed intelligence on the edge-to-cloud continuum: A systematic literature review," *Journal of Parallel and Distributed Computing*, 2022.

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

[6] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[7] M. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 8, pp. 1–37, 2021.

[8] N. K. Alham, M. Li, Y. Liu, and M. Qi, "A mapreduce-based distributed svm ensemble for scalable image classification and annotation," *Computers & Mathematics with Applications*, vol. 66, no. 10, pp. 1920–1934, 2013.

[9] F. Özgür Çatak and M. Erdal Balaban, "A mapreduce based distributed svm algorithm for binary classification," *arXiv e-prints*, pp. arXiv–1312, 2013.

[10] Q. Chen and F. Cao, "Distributed support vector machine in master–slave mode," *Neural Networks*, vol. 101, pp. 94–100, 2018.

[11] J. Dass, V. Sarin, and R. N. Mahapatra, "Fast and communication-efficient algorithm for distributed support vector machine training," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 5, pp. 1065–1076, 2018.

[12] J. Chen, K. Li, Z. Tang, K. Bilal, S. Yu, C. Weng, and K. Li, “A parallel random forest algorithm for big data in a spark cloud computing environment,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 919–933, 2016.

[13] Y. Zhang, Z. Xiong, J. Mao, and L. Ou, “The study of parallel k-means algorithm,” in *2006 6th World Congress on Intelligent Control and Automation*, vol. 2. IEEE, 2006, pp. 5868–5871.

[14] W. Zhao, H. Ma, and Q. He, “Parallel k-means clustering based on mapreduce,” in *IEEE international conference on cloud computing*. Springer, 2009, pp. 674–679.

[15] M.-F. F. Balcan, S. Ehrlich, and Y. Liang, “Distributed k-means and k-median clustering on general topologies,” *Advances in neural information processing systems*, vol. 26, 2013.

[16] J. Qin, W. Fu, H. Gao, and W. X. Zheng, “Distributed k-means algorithm and fuzzy c-means algorithm for sensor networks based on multiagent consensus theory,” *IEEE transactions on cybernetics*, vol. 47, no. 3, pp. 772–783, 2016.

[17] M. M. A. Patwary, D. Palsetia, A. Agrawal, W.-k. Liao, F. Manne, and A. Choudhary, “A new scalable parallel dbscan algorithm using the disjoint-set data structure,” in *SC’12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE, 2012, pp. 1–11.

[18] M. Götz, C. Bodenstein, and M. Riedel, “Hpdbscan: highly parallel dbscan,” in *Proceedings of the workshop on machine learning in high-performance computing environments*, 2015, pp. 1–10.

[19] M. Chen, X. Gao, and H. Li, “Parallel dbscan with priority r-tree,” in *2010 2nd IEEE International Conference on Information Management and Engineering*. IEEE, 2010, pp. 508–511.

[20] G. Luo, X. Luo, T. F. Gooch, L. Tian, and K. Qin, “A parallel dbscan algorithm based on spark,” in *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud)*. IEEE, 2016, pp. 548–553.

[21] C. Pizzuti and D. Talia, “P-autoclass: scalable parallel clustering for mining large data sets,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 3, pp. 629–641, 2003.

[22] “Apache Mahout,” https://mahout.apache.org//, accessed June 2022.

[23] “Apache Spark’s MLlib,” https://spark.apache.org/mllib/, accessed June 2022.

[24] N. e. a. Kukreja, “Training on the edge: The why and the how,” in *2019 IEEE International Parallel and Distributed Processing Symposium Workshops*. IEEE, 2019, pp. 899–903.

[25] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, “When edge meets learning: Adaptive control for resource-constrained distributed machine learning,” in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 63–71.

[26] Y. Zhang, M. J. Wainwright, and J. C. Duchi, “Communication-efficient algorithms for statistical optimization,” *Advances in neural information processing systems*, vol. 25, 2012.

[27] A. Kumar, S. Goyal, and M. Varma, “Resource-efficient machine learning in 2 kb ram for the internet of things,” in *International Conference on Machine Learning*. PMLR, 2017, pp. 1935–1944.

[28] C. Gupta, A. S. Suggala, A. Goyal, H. V. Simhadri, B. Paranjape, A. Kumar, S. Goyal, R. Udupa, M. Varma, and P. Jain, “Protonn: Compressed and accurate knn for resource-scarce devices,” in *International Conference on Machine Learning*. PMLR, 2017, pp. 1331–1340.

[29] M. T. Yazici, S. Basurra, and M. M. Gaber, “Edge machine learning: Enabling smart internet of things applications,” *Big data and cognitive computing*, vol. 2, no. 3, p. 26, 2018.

[30] F. Nan, J. Wang, and V. Saligrama, “Pruning random forests for prediction on a budget,” *Advances in neural information processing systems*, vol. 29, 2016.

[31] H. H. Kumar, V. Karthik, and M. K. Nair, “Federated k-means clustering: A novel edge ai based approach for privacy preservation,” in *2020 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*. IEEE, 2020, pp. 52–56.

[32] D. K. Dennis, T. Li, and V. Smith, “Heterogeneity for the win: One-shot federated clustering,” in *International Conference on Machine Learning*. PMLR, 2021, pp. 2611–2620.

[33] R.-H. Hsu, Y.-C. Wang, C.-I. Fan, B. Sun, T. Ban, T. Takahashi, T.-W. Wu, and S.-W. Kao, “A privacy-preserving federated learning system for android malware detection based on edge computing,” in *2020 15th Asia Joint Conference on Information Security (AsiaJCIS)*. IEEE, 2020, pp. 128–136.

[34] Y. Wu, S. Cai, X. Xiao, G. Chen, and B. C. Ooi, “Privacy preserving vertical federated learning for tree-based models,” *arXiv preprint arXiv:2008.06170*, 2020.

[35] H. Yao, J. Wang, P. Dai, L. Bo, and Y. Chen, “An efficient and robust system for vertically federated random forest,” *arXiv preprint arXiv:2201.10761*, 2022.

[36] Y. Han, P. Du, and K. Yang, “Fedgbf: An efficient vertical federated learning framework via gradient boosting and bagging,” *arXiv preprint arXiv:2204.00976*, 2022.

[37] Y. Liu, Y. Liu, Z. Liu, Y. Liang, C. Meng, J. Zhang, and Y. Zheng, “Federated forest,” *IEEE Transactions on Big Data*, 2020.

[38] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, “A survey of machine learning for big data processing,” *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1, pp. 1–16, 2016.

[39] C. Savaglio, P. Gerace, G. Di Fatta, and G. Fortino, “Data mining at the iot edge,” in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2019, pp. 1–6.

[40] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith, M. J. Franklin, and M. I. Jordan, “Mlbase: A distributed machine-learning system.” in *Cidr*, vol. 1, 2013, pp. 2–1.

[41] L. Belcastro, R. Cantini, F. Marozzo, A. Orsino, D. Talia, and P. Trunfio, “Programming big data analysis: Principles and solutions,” *Journal of Big Data*, vol. 9, no. 4, 2022.

[42] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” *arXiv preprint arXiv:1610.05492*, 2016.

[43] G. Jagannathan and R. N. Wright, “Privacy-preserving distributed k-means clustering over arbitrarily partitioned data,” in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005, pp. 593–599.

[44] M. S. Aslanpour, S. S. Gill, and A. N. Toosi, “Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research,” *Internet of Things*, vol. 12, p. 100273, 2020.

[45] D. Talia, P. Trunfio, and F. Marozzo, *Data Analysis in the Cloud: Models, Techniques and Applications*. Elsevier, October 2015, iSBN 978-0-12-802881-0.

[46] F. Tao, D. Zhao, Y. Hu, and Z. Zhou, “Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system,” *IEEE Transactions on industrial informatics*, vol. 4, no. 4, pp. 315–327, 2008.

[47] S. GiampÃ , L. Belcastro, F. Marozzo, D. Talia, and P. Trunfio, “A data-aware scheduling strategy for executing large-scale distributed workflows,” *IEEE Access*, vol. 9, pp. 47 354–47 364, 2021, iSSN: 2169-3536.

[48] A. Barbieri, F. Marozzo, and C. Savaglio, “Iot platforms and services configuration through parameter sweep: a simulation-based approach,” in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 17-20 October 2021, pp. 1803–1808.

[49] M. Sinqadu and Z. S. Shibeshi, “Performance evaluation of a traffic surveillance application using ifogsim,” in *International Conference on Wireless Intelligent and Distributed Environment for Communication*. Springer, 2020, pp. 51–64.

[50] L. Belcastro, A. Falcone, A. Garro, and F. Marozzo, “Evaluation of large scale roi mining applications in edge computing environments,” in *25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, 2021, pp. 1–8.

[51] L. U. Khan, I. Yaqoob, N. H. Tran, S. A. Kazmi, T. N. Dang, and C. S. Hong, “Edge-computing-enabled smart cities: A comprehensive survey,” *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 200–10 232, 2020.

[52] Q. Wang, X. Zhu, Y. Ni, L. Gu, and H. Zhu, “Blockchain for the iot and industrial iot: A review,” *Internet of Things*, vol. 10, p. 100081, 2020.

[53] S. Tian, W. Yang, J. M. Le Grange, P. Wang, W. Huang, and Z. Ye, “Smart healthcare: making medical care more intelligent,” *Global Health Journal*, vol. 3, no. 3, pp. 62–65, 2019.