Review of Algorithms, Frameworks and Implementation of Deep Machine Learning Algorithms

[1]Ivan Leonid
[1]Faculty of Computer Science, HSE University, Moscow, Russia.
[1]leoniddss@hotmail.com

Correspondence should be addressed to Ivan Leonid: leoniddss@hotmail.com.

Article Info
Journal of Computing and Natural Science (http://anapub.co.ke/journals/jcns/jcns.html)
Doi: https://doi.org/10.53759/181X/JCNS202202016
Received 20 May 2022; Revised form 18 June 2022; Accepted 25 July 2022.
Available online 05 October 2022.

Abstract – Machine Learning (ML) is increasingly being used in intelligent systems that can perform Artificial Intelligence (AI) functions. Analytical model development and solving problems related with it may be automated by machine learning, which explains the ability of computers to learn from problem-specific learning algorithm. Depending on artificial neural networks, "deep learning" is a kind of machine learning. The performance of deep learning techniques is superior to that of superficial machine learning techniques and conventional methods of data analysis in many situations. Deep Machine Learning (DML) algorithms and frameworks that have been implemented to and supported by wireless communication systems have been thoroughly analyzed in this paper. User associations, power latency and allocation; bandwidth assignment and user selections, and; cloud computing technology on the edge have both been suggested as potential DML implementations.

Keywords – Deep Machine Learning (DML), Federated Learning (FL), Machine Learning (ML).

I. INTRODUCTION

Data collecting has increased at an unprecedented rate as a result of the fast evolution of new innovations in recent years. Because of the problem's complexity, Machine Learning (ML) methods are increasingly being employed to evaluate datasets and develop decision-making systems. Self-driving automobiles can be controlled, voice recognition can be used, and customer behavior may be predicted, for example. As terabytes of data are necessary to train models for complex applications, solution designers are often encouraged to employ distributed systems to potentially advance parallelism and the overall input/output bandwidth due to the lengthy training times. Whenever data is large and more scattered to store in single system, a more centralized strategy is not even the option to consider. In many firms, the transaction processes on data stored in different location or the astrophysical data, which is larger to consolidate and relocate are some of the options.

In recent years, ML technology has proliferated in more and more complicated applications. Despite the proliferation of different methodologies and algorithms, the underlying data structures are very similar. There are a lot of well-known issues in linear algebra that make up the vast bulk of the computation in ML techniques. For decades, researchers in the field of high-performance computing have been working to find ways to improve these kinds of processes. This has resulted in the effective adoption and integration of several methods and tools from the HPC domain (e.g., BLAS or MPI). The HPC community has also recognized machine learning as a developing high-value task and begun applying HPC techniques to it. It took Strategy [1] only three days to train a 1 billion variable system on their COTS HPC machine, which they purchased off the shelf. A neural network was trained on Intel's Knights Landing, a processor built for high-performance computing (HPC) applications. It has been shown that deep learning applications like extracting weather patterns may be tuned and scaled effectively on parallel plate HPC systems by Nandal [2].

For example, Fan and Zhang [3] used HPC approaches like lightweight profiling to forecast the workload requirement for rescheduling deep neural network applications on cloud computing environment. Resilience features of Deep Neural Networks (DNN) while operating on injectors, which are often used in large HPC systems, were studied by Zhu and Zhao [4]. For large-scale computing issues, there are two inherently independent and complimentary techniques of accelerating workloads: vertical scaling (or scaling up) and adding additional nodes to the network (horizontal scaling). This paper evaluates the algorithms and frameworks that have been implemented to and supported by wireless communication systems. Based on this rationale, this paper has been organized as follows: Section II focusses on the DML definition, historical and scientific foundation. Section III reviews the past literature works regarding the algorithms, and frameworks of DML. Section IV analyses the algorithms and frameworks of DML, while Section V focusses on the frameworks of DML. Section VI reviews the implementation of DML, and lastly, Section VII draws finals remarks about the paper.

II. DML DEFINITION, HISTORICAL AND SCIENTIFIC FOUNDATION

Definition

The term “distributed machine learning," refers to algorithms and systems with several nodes, which are purposed to boost performance, enhance accuracy and deal with massive data. Big data may lower the learning error of many algorithms, making it more efficient than utilizing more sophisticated approaches. It is possible to extract useful conclusions from a big quantity of data using Distributed Machine Learning (DML). Machine Learning (ML) activities may be performed in a distributed setting using several technologies. Database, generic, and purpose-built systems are three of the most typical forms of frameworks evident in the modern world. Each kind of system has different benefits and drawbacks, but they are all employed in reality depending on specific performance needs, use cases, input data amounts and degree of implementation work required.

Historical Analysis

Distribution frameworks have replaced the need for handwritten systems in which the user had to actively manage all parts of the operation. Data distribution, parallelism, synchronization, and fault tolerance were all part of this time-consuming, prone to mistake process, which made it difficult for users to troubleshoot current algorithms or create new ones. Novel programming models, such as MapReduce, have simplified distributed computing and made it possible for individual users to scale an algorithm to big data. The concepts of file systems, distributed runtime setting and parallel process are provided by these technologies, enabling users to concentrate on developing methods rather than maintaining low-level details.

Scientific Fundamentals

In order to run ML approaches within a distributed environment, users should start with conceptually transforming the single-threaded approaches to parallel approaches. If the user doesn't have a good grasp of the underlying method, this might be the most difficult part of the process Second, the parallel algorithms must be implemented. To do this correctly and efficiently, the user must have a thorough understanding of the system's semantic and runtime. Section IV provides a critical review of the Deep Machine Learning (DML) algorithms. However, Section III starts by reviewing the works about ML implementation in communications and networking, as well as WSN.

III. LITERATURE REVIEW

Works on ML in Communications and Networking

Ma, Liu, Cao, Zhu and Liu of [5] have investigated a few examples of how ML is being used in networking and communications in this article. Use of machine learning to tackle wireless channel state challenges is a major contribution from these approaches. Though these studies have showed considerable performances improvement on their assessment, other works merely concentrate\son trace-driven simulations or testing utilizing the same data sets used for learning rather than genuine real-world experiment. This study is important because it cleared the path for future models to be built that can accurately anticipate the future status of wireless channels based on numerous items of data.

Signal Classification

A sender's signal must be appropriately detected by a receiver in order to execute dependable wireless communication. Disturbance from transmitted signal and actual wireless link settings are important considerations for the Mallick, Dhara and Rath of [6]. Machine learning can be used to categorize signals in a realistic wireless channel, according to the researchers. Because it identifies a signal that has been altered as it travels across a communication network, we liken this process to pattern identification.

Traffic Classification and Data Collection for Network Management

The Benzekki, El Fergougui and Elbelrhiti Elalaoui of [7] underline the necessity of understanding the sort of data that may be gathered in SDNs and the process of learning knowledge from that data. Using an OpenFlow-based architecture implemented in a business network, this research takes a first milestone toward machine learning-based network control. Nevertheless, this research just focuses on OpenFlow-based traffic monitoring and categorization, and does not propose a complex ML-based system or network management. But this study has prepared the road for the application of ML-based networks administration and demonstrated basic instances of ML approaches being used in.

Network Attack Forecast

SDN controller security policies may be defined using machine learning. Security protocols on the SDN console are established to prevent the access of possible adversaries by blocking the whole subnetwork, using machine learning techniques. The same datasets were used for both training and testing purposes in the assessment of the suggested approach.

Wireless Adaptive Streaming

Over time, network parameters may change dramatically, and this is true even in the same setting. As a result, it is difficult to forecast the state of the network in the future. Several rate-adaptation algorithms have been developed for high-quality video streaming; however, they are not adequate and there is still potential for development. Video streaming services might

benefit from the use of machine learning methods, according to a number of studies. Proposed a system that is installed on the server-side, learns essential characteristics, and makes the optimal choice on bitrate and CDN for the streaming client to improve QoE. The optimum ABR algorithm may also be generated via reinforcement learning, which takes into account the bandwidth, buffering level, and stream rate.

Mobile Cloud Offloading

When it comes to mobile device compute offloading, cloud computing is a viable option. To determine if computation should be offloaded to the cloud it monitors device networks and information conditions. Network data such as throughput is only one of several factors considered when making a choice on cloud computing in [8], which uses machine learning. In contrast to the values impacted by a volatile and uncertain wireless connection, other input data include user input, device power level, and CPU usage level.

Works on ML in WSN

Machine learning methods have been used to sensor networks in a variety of ways. These approaches also have a significant impact in terms of using machine learning to address hard challenges created by lossy networks and limited devices. As a result of these studies, it is now possible to develop credible models that can accurately forecast the future configuration of lossy networks.

Channel Error Diagnostics

IEEE 802.11, 802.15.1 and 802.15.4 all use the ISM band. As a result, the effectiveness of communications is degraded by interfering systems. It was discovered by Phuah et al. of [9] that there are distinct pattern variations for main wireless situations in their investigation of IEEE 802.15.4's error frequencies. This discovery led them to develop a machine learning method for classifying wireless channel faults into several subcategories and to develop a system for diagnosing various problems in IoT networks.

Spectrum Decision

The work of Solares, Sboui, Rezki and Alouini [10] also mentions ISM band pollution as well as sensor node power constraints. To address this poor setting, it presents machine learning method for channel selection. The algorithm forecasts a number of probable transmissions attempt using machine learning techniques. RSSI, number of communications tries, reasons for each unsuccessful attempt, and key metrics such as and LQI and RSSI from the previous received packet are used as input data. It chooses the best route from the output, and a channel with a low amount of projected transmission tries is deemed preferable.

Outlier Detection

Because a wireless sensor network with restricted nodes is prone to interference, unstable channels, and cyber-intrusion, systems performance suffers, and fraudulent data may be sent to higher levels of management. This might pose serious issues with sensor communication networks used in public safety and industrial automation. Wang, Caja and Gómez of [11] note out that present outlier identification methods demand a lot of memory, a lot of computes, a lot of energy, a lot of communication overhead, and they do not handle extensive online data flow. To overcome the issue, they suggested employing a multi-agent paradigm to identify online outliers deploying a machine learning method.

Indoor Localization

In general, one of the most well-known instances of object localisation is GPS. Due to the low transmitted GPS signal strength, it is difficult to determine the accurate position of an item within a structure. As a result, several techniques are applied. For instance, several endpoints are used as connection points, and this data is being used to approximate a target object's relative position. It's critical to develop an accurate indoor localization system since it may be utilized to improve safety in subterranean mines or caverns. Nevertheless, disturbance on the wireless medium still remains, reducing estimate accuracy. To solve the challenge, examined the performance of seven different ML algorithms on two distinct architectures to discover the solution with the fewest mistakes. The person on the testbed wore a wearable sensor that allowed him to locate himself inside the sensor network.

Event Detection

Various applications employ wireless sensor networks. The research focuses on pipeline leak detection in the water and oil transportation infrastructure. It employs a pattern detection algorithm to teach the sensor system to recognize and categorize fresh traces of events, such as leaks. Distributed sensor nodes work together to determine the magnitude of the leakage event. Despite the fact that this project contains a wireless sensor network, the challenges that come with employing an unreliable and risky wireless channel were hardly explored.

Fault Detection

Various issues caused by a susceptible and fluctuating wireless medium, as well as inexpensive sensors, result in incorrect data obtained from the sink node. To swiftly respond and maintain the infrastructure, fault information should be noticed and the source of the occurrence should be determined. On the basis of ML, the study established a statistical strategy to detect and diagnose defects in a wireless sensor network. It divided faults into two types: data faults and system faults. Faults induced by a deteriorated or failing sensor are classed as data faults, whereas system errors are triggered by low battery, calibrations, communications, or connection problems. Chaudhari of [12] used machine learning to investigate defect detection in a similar context. They divided faults into 4 groups: offset faults, gain faults, stuck-at faults, and out of limits faults.

Routing

Multi-hop routing system may be made more energy effective by using ML techniques. Krishna, Niranjan and Shireesha [13] suggested an ML-based clustering strategy for efficiently assigning sensors to the closest cluster. Kumar Kamila, Dhal and Nayak [14] employed machine learning to improve the routing mechanism in a WSN. The envisaged routing algorithm aims to extend lifetime of the network and transfer information packets in the shortest possible time. These studies claim that using machine learning methods on WSN may help with resources planning.

Supervised and unsupervised methods for machine learning are the two main subcategories of machine learning techniques. Unlabeled inputs (e.g., an email marked spam/non-spam) may be predicted using a model built from labeled inputs (e.g., a batch of labeled emails). For example, clustering consumers for market research using unsupervised learning seeks to uncover attributes about the dataset without depending on labeled examples. In this paper, a critical analysis of DML algorithms and frameworks will be evaluated. Section IV focus on the DML algorithms.

IV. DML ALGORITHMS

Parallelization parameters for gradient descent, regression, k-means clustering, and ensemble learning are presented in the next section. Consider the following examples of algorithms and approaches, but keep in mind that there are several alternative approaches that may be used in the real world.

Gradient Descent

There are several algorithms for minimizing a loss function f in machine learning, including gradient descent. ML operations e.g., Support Vector Machine (SVM), and linear regression may be performed with this loss function. linear regression loss functions include the Mean Squared Error (MSE) that evaluates the mean variation between the observed and forecasted values over the various training sets. The findings of the gradient descent approaches are the vectors denoted by θ, typically identified as a weight vector or model that integrates the loss function coefficient, which best fits the training dataset. Unsupervised input data may be predicted using a model that has been built. Traveling in the path of the loss function's highest negative gradient updates the weight vector repeatedly as gradient descent starts.

As indicated in **Algorithm 1**, the weight trajectory θ_{j+1} denoted by has been upgraded by visualizing the current weight element denoted by θ_j and considering subtraction of the loss function gradient evaluated with the prevailing mode where, f is measured the training rate. When using batch gradient descent, metrics are transformed after the training sets denoted by n have been processed. Probabilistic gradient descent is substantially more versatile than batch gradient descent since it changes model weights for each randomized training occurrence. There is evidence that the stochastic gradient descent method converges more quickly than batch gradient descent because model changes are made instantly for each example. A single run over the training data may often provide optimum model parameters using stochastic gradient descent.

Algorithm 1: The Batch gradient descent

Random initialized
whereas! converged do.
$\theta_{j+1} = \theta_j - \alpha \nabla f(\theta_j)$
end while

Regression

Logistic and linear regression alludes to the discriminative classification approaches that make use of the labeled training sets in order to identify the hyperplanes w, which separates two different groups of datasets. Categorically, given the training sets, every form $f \times 1$; $f \times 2$; and so on; x_m; y_g, with x_i the feature value and y as the binary label, the technique is able to determine the co-efficient θ of the logistic or linear function, which fits the training set. Once it has been constructed, the model can be employed to group unlabeled datasets. When dealing with linear regression, it is possible to get a closed-form equation using an Ordinary Least Square (OLS) approach. Provided the collection of the x feature vector and the collection of the y corresponding labels, θ could be computed by employing the following expression:

$$\theta = \lfloor x^t x \rfloor^{-1} x^t y \quad (1)$$

Even though the closed-form equation in **Eq. 1** is computed with ease based on the application of the standard linear algebra strategies, the matrix inversion is intensive and not grading to many training cases. When doing regression assignments in a distributed environment, the previously discussed techniques centered on gradient descent are commonly employed. Locally calculate and store model changes are done utilizing a disjoint sample of data input by each concurrent worker in this scenario. After every user has finished its localized transformations to the system, the alteration are interlinked internationally and potentially dispersed to the upcoming iteration.

K-Means Clustering

A machine learning approach known as K-means clustering divides input data into k groups in an unsupervised iterative fashion. Based on these initial cluster centers, the nearest input data point is assigned using K-means. In order to calculate the new center value once all data sets have been allocated, the method averages each centroid's feature values. After a certain number of iterations, the algorithm stops until specific convergence requirements are met. In a distributed situation, k-means clustering may be performed utilizing data-level form of parallelism whereby every computing node is acting on a different set of disjoint data sets. All present centroids are computed and the nearest centroids to every local data element are determined in this situation. Next, the method sets sumc D sumc C d and countc D countc C 1 for a particular item of data denoted by d assigned to the centroid denoted by c. A global total and count of all compute nodes' local input data objects is then computed to get the new central values.

Ensemble Learning

To improve overall classification accuracy, ensemble learning involves developing a diverse set of classifiers. Multiple weak learners may be integrated into a single strong learner that is more accurate than any one weak learner could be on its own. Weak learners may be trained in parallel based on the application of a smaller subset of the input dataset to employ ensemble approaches in many contexts. You may build a single strong learner to classify unlabeled input dataset by training of the n classifier whereby n is consider the overall listing of distributed clients, in one of two approaches. To create a single classifier from all n classifiers, the first procedure may be used. Using this method is easier for classifier ensembles, which all utilize similar learning techniques; however, this is not always the case for those that use different algorithms (e.g., regression models against decision trees). There are two ways to combine n scattered classifiers: using the datasets from every classifier in a separate form. Bootstrap and bagging aggregation, is a technique for merging several classifier results. Bootstrap aggregation is used to pick the mode of the n output values and create the final output value.

V. DML FRAMEWORKS

Several systems may be able to handle Distributed Machine Learning (DML) applications. DML frameworks are grouped into three classes: purpose-built, generic and database. There are a number of ways in which traditional Database Management Systems (DBMSs) may be utilized to build and implement machine learning tasks. A DBMS must be modified or extended in order to execute most ML methods operations because SQL protocol does not effectively express various aspects of the methods (such as iteration). The systems below may be used to do machine learning operations directly in a Database Management System (DBMS). Unified architecture is used to build a gradient descent method by Zhao, Zhang, Zhou, Chen, Jin and Liu [15]. A user-defined aggregate objective function can now be used with the new gradient descent operator to implement new machine learning algorithms. SQL extensions are not the only thing that can be done with some systems, such as MADlib. The following query may be used to do a logistic regression within the present database:

```
Choose madlibs.logregr_train (
source_tables,
out_tables,
dependent_varnames,
independent_varnames,
grouping_colss,
max_iters,
optimizers,
tolerances,
verboses
)
```

This kind of solution allows the use to complete ML activities on data, which has been recorded onto the BMBS, ignoring the requirement to potentially transfer datasets into alternative systems. When it comes to transforming data, however, users are typically constrained to a set of procedures that are predetermined.

General Frameworks

Data processing workflows may be developed in a host language using a set of API operators provided by standard frameworks. Many of these frameworks come with pre-installed machine learning algorithms since they are scalable and can manage any demand. Many features are available, from automatic fault tolerance to a configurable API, in even the most

sophisticated and high-level frameworks. The Message Passing Interface (MPI) alludes to the lower-level foundation for higher performance distributed computation. The various primitives provided by MPI (e.g., scatter/gather, spread/broadcast, receive/send) may be used to create machine learning algorithms. As a result of the low-level nature of MPI, machine learning projects executed with it are often time-consuming and error-prone.

Commodity machine clusters may be utilized to run Hadoop, one of the most popular MapReduce implementations. Hadoop's fault tolerance, simple programming abstraction and distributed file system make it probably to evaluate petabyte-scale dataset over various computers. In order to use Hadoop for iterative operations, each iteration must be submitted as a distinct job. In addition, since intermediate findings should be copied onto the disk, iterative queries are less efficient. Rather than forcing customers to change their techniques to fit the MapReduce paradigm, Mahout provides Hadoop distributed runtime implementations of different ML workloads.

Spark alludes to a distributed model for the in-memory computing, which employs API operators given in the Python, Java and Scala. Iterative queries and other tasks may be carried out in parallel thanks to the use of driver software, which keeps everything in sync. Spark's extra operators, which go beyond the core map-reduce set and provide more descriptive operators like filter, join, and unison, simplify the explanation of many machine learning techniques. Since Spark stores the working dataset in memory, it also supports iterative approaches.

LINQ applications may now be executed on the Dryad distributed runtime thanks to Microsoft's DryadLINQ. While LINQ is a high-degree language for manipulation of data, Dryad alludes to the execution engine, which defines the tasks as a dataflow graph. Numerous machine learning-related things may be found in both. It is challenging to do complex calculations, such as ML, by using in-memory analytics, such as Tupleware. Tupleware compiles the procedures scripted in LLVM-centered programming languages, unlike other systems, which do not. By directly compiling workflows with Tupleware, rather than having to deal with the significant overhead that comes from interpreting execution models, many optimizations can be applied to code generation that take the properties of input dataset, underlying hardware and user-defined computations into account.

Purpose-Built Frameworks

Specialized ML systems have been established as a result. For instance, the system either give domain-based ML terms or algorithm-based development, which are not globally applied. SystemML alludes to a high-degree and prescriptive language, which could be employed to structure ML tasks. There are several built-in matrix operators accessible in this R-like syntax language. Iterative runs on the data are minimized by turning MapReduce jobs into workflows that the system can execute. OptiML provides a domain-based and embedded language developed on a linear algebraic expression. Graph, matrix and vector data forms are all available, including the subsets, which could be employed to effectively optimize systems. The systems produce codes for special hardware (e.g., GPU – Graphics Processing Unit) and the multi-core computers.

Hogwild! Alludes to the lock-free applications of stochastic gradient descent. The approach eradicates any locks because all of the framework data is encrypted in share memories. Permits other processors to visualize the most recent versions of the framework right away, avoiding the enormous cost of locking. Removing locking resulted in lower error rates and faster convergence, according to the authors. It is possible to use the Columbus paradigm for feature selection in many analytics frameworks. Sheikh [16] gives a collection of algorithms and associated improvements for determining the most suitable features for a ML issue. A high-degree declarative language called MLbase allows users to specify machine learning tasks to the system. Each learning task is assigned a unique set of parameters and processes by the system's optimizer (for example, classify a particular dataset). A final model and data summary are shown to the user as a result of this procedure.

VI. KEY DML IMPLEMENTATIONS

Many applications of machine learning may be found in the fields of image analysis, spam detection, recommendations systems, bioinformatics, and natural language processing. The amount of data available in almost every application is increasing, and users require an efficient and simplified approach to evaluate these distinct data sources, Deep Machine Learning (DML) approaches allow for the extraction of conclusions from big datasets in a reasonable amount of time. Complexity and delayed convergence to (sub-)optimal solutions in massive wireless systems with big data could be caused by dynamic programming, convex optimization and gradient descent-based methods. Using data-driven ML techniques, problems may be solved more rapidly, but a device that has more powerful computing and storage capacity than typical mobile devices is required. This restriction may be overcome by centrally training an ML algorithm using a cloud or edge unit that collects data from all wireless devices. In addition, bandwidth, channel conditions, latency, energy consumption, and privacy issues further restrict this approach. Federated Learning (FL) and Partitioned Learning (PL) have been established to permit wireless gadgets acquired a globalized framework with constrained data transformation or on the foundation of partial datasets and models.

Power Control

In [17], a novel cell-free massive multiple-input multiple-output (CFmMIMO) system framework has been proposed.

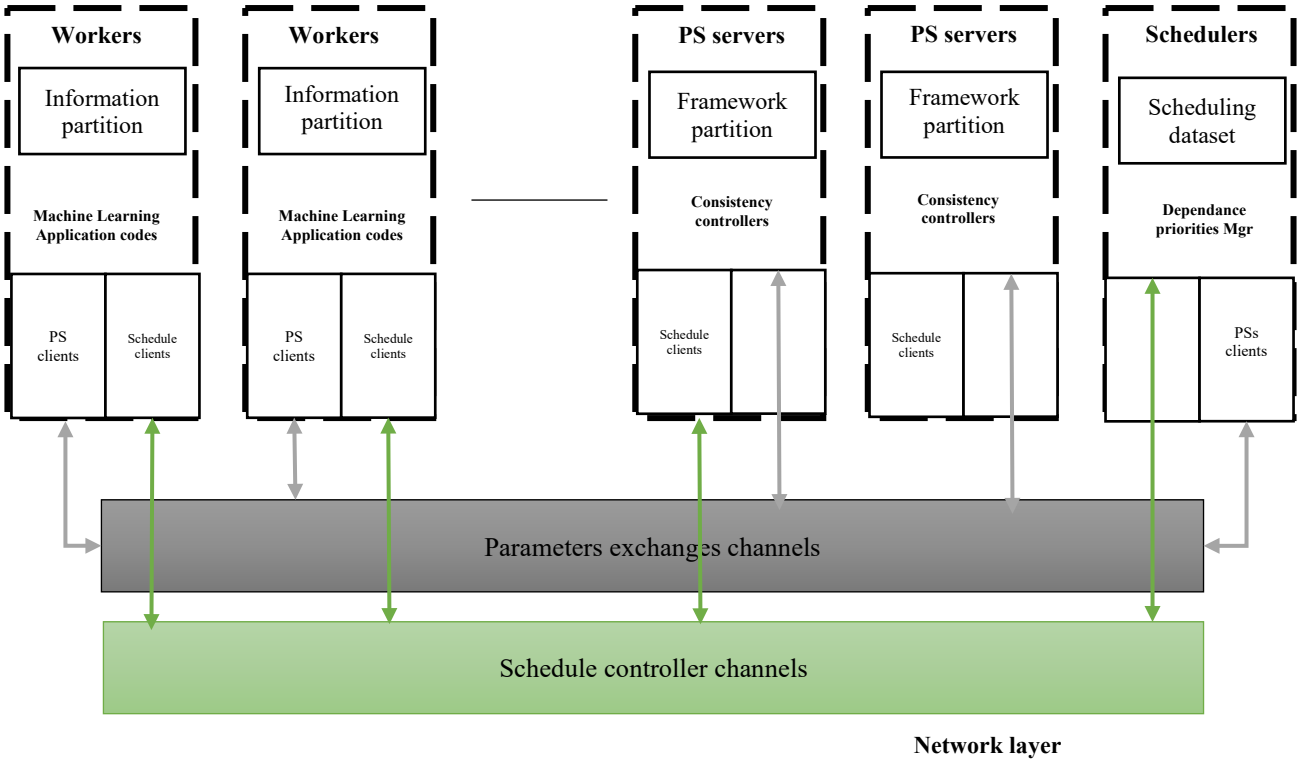

Fig. 1: Petuum system showing schedulers and PS clients

There is scheduling, clients, and parameter servers are shown in **Fig. 1** illustrating the Petuum systems. The decentralized and shareable memory of the systems may be accessed from any node using a simple interface provided by the central system's parameter server model. For the ML model to be stable, the system additionally measures the inconsistency that arises from asynchronous computations. Each FL repeat must take place inside a long coherence period if it is to be stable. Nonconvex mixed-timescale stochastically nonconvex problem of power management for optimizing user throughput, local accuracy, and transmit power is described. Training, communication, and mathematical operations of weight replenishment are all captured in this interoperation.

Utilizing a digital sequential convex estimation approach, the power control issue is repeatedly summoned with the guaranteed convergence to within some dimension of stationary resolutions. Simulations indicate that the hybrid optimization limits the learning timeline by approximately 50 percent compared to different benchmark methods, e.g., i) equalized downloading power assignment to all the user tools with the high uploading transmitting fixed localized accuracy and power; ii) equal power allocation with highest uploading transmission power, and iii) equalized power assignment with uploading and downloading transmission power. In contrast to big MIMO as well as TDMA, including the located MIMO, CFmMIMO has been shown to need the least amount of FL training time. This technique may be limited by the assumption of time-invariant channels state data' however, the conclusions are still helpful.

For a stable and reliable AirComp over fading channels, edge device (or agent) power management is critical. Traditional data gathering generally assumes that data gathered locally by numerous agents is the same. This is not always true. The data is standardized using a Gaussian process with zero mean and one standard deviation. For the purpose of collecting gradients in ML, it is incorrect to assume that the gradient distribution is constant throughout iterations. Ovtchinnikov [18] employ gradient statistics for over-the-air Federated Learning (FL), as shown in **Fig. 2**. When gathering data for the models, it is critical that the Mean Square Error (MSE) be as near to zero as feasible. To do this, the transmit power and edge server de-noising coefficient of each agent are pooled together in a cooperative fashion. An optimal solution may be obtained in closed form if the gradient's first and second order statistics are known ahead of time.

Multivariate gradient variation parameters are utilized to determine the optimal solution. It is based on past observations to estimate the statistical features of the gradient. These data will be used in subsequent cycles to modify the transmit power of devices. The power management approach outfits the basis complete-power communication approach outfits a basic complete-power communications approach and the threshold-centric power management technique based on convergence speed and model correctness. They build a FL communication infrastructure with relays to facilitate model update, transmission, and trade, as described in [19]. Mobile devices generate models upgrades inside the platform based on locally collected or generated sample data. A cooperative relay network may be used to communicate these modeling updates to a master model. Training tasks provided by mobile devices are advantageous to the model master; in exchange, the model master pays fees to use these services.

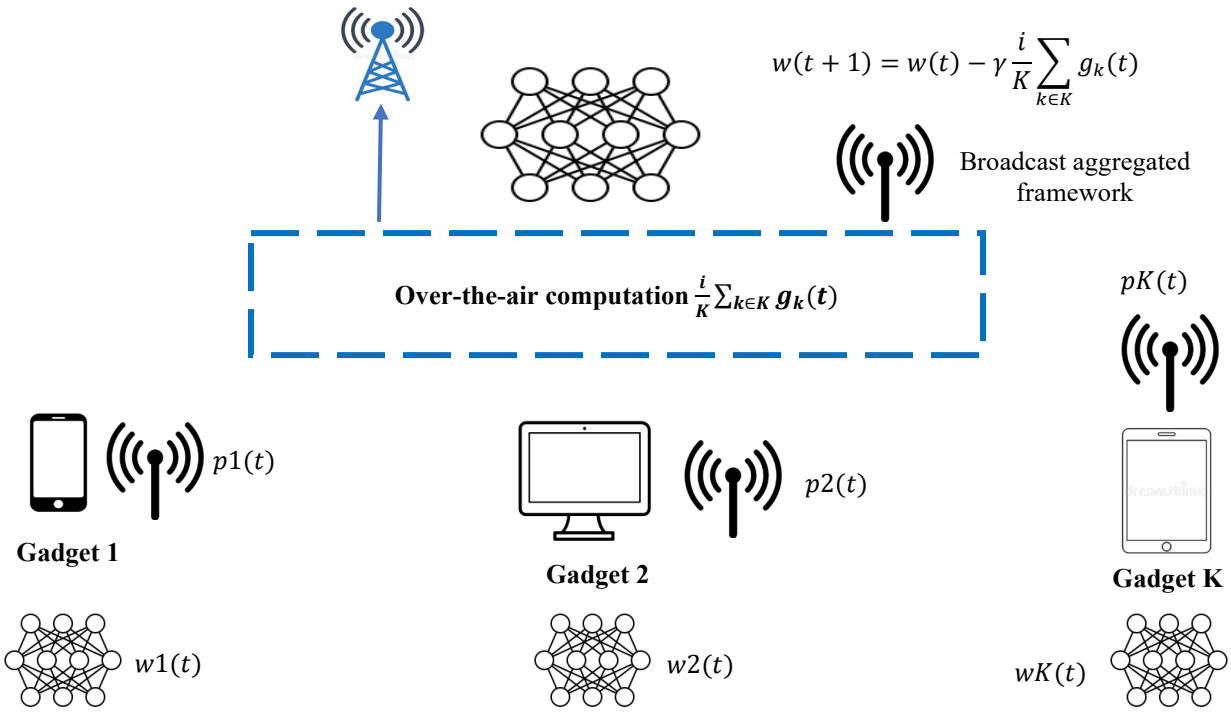

Fig. 2: A depiction of a federated learning (over-the-air)

where $k \in \{1,\cdot,K\}$ signifies the devices' index at the edge node, w alludes to the model matrix, p signifies the transmission power, while γ represents the training, and $gk\ (t)$ alludes to the loss function's gradient that is computed at gadget k at a specific time t. In order to avoid interfering with each other's wireless transmissions, it is essential that each device use a distinct relay node and set a transmit power. A Stackelberg game theory [20] aims to explain the reciprocal actions of the units and the modeling master's interactions with the devices. The outside point approach is used to obtain Stackelberg equilibrium.

Spectrum Control

Edge ML approaches might take advantage of computational resources and big distributed data. FEEL infrastructure has received a lot of attention due to its intention to secure users' data. FEEL schedules the general model training of every server, including the local model learning of every edge agent, as a segment of the whole model-learning process. In [21], bandwidth distribution and coordination are improved for the most efficient usage of mobile devices.

The training impact is ensured and the total energy consumption is reduced by tailoring the optimal procedures to the device's channel statuses and compute capabilities. A smaller spectrum is assigned to those with stronger channels in conventional throughput-maximizing systems, while a wider spectrum is used optimally for those with weaker processors. FEEL's simultaneous model renewals have the unpleasant side effect of unfavorable channel conditions and insufficient computations. Weighing factors such as channel power and complicated mathematical computations are used to reward agents who put in more effort and effort. The experiments' results show that significant energy reductions are feasible. Due to a system constraint, the system cannot be used for additional framework transmission cases, e.g., asynchronous frameworks.

A considerable number of mobile users in the radio coverage dimension of BS use micro-BS or pico-BSs to conduct FL. As part of a network-wide agreement, the BSs broadcast their model renewals to the macro-BS on a regular basis. By integrating gradient sparsification with periodic averaging, the hierarchical FL may communicate more effectively. Using a sparse gradient vector, which sends just a part of the parameters each time, it is feasible to decrease communication latency. There is a total of 60, 000 32 by 32 color pictures in the CIFAR-10 dataset that are broken down into 10 categories and 6, 000 images in each. There are tens of thousands of photographs in the total of 50, 000 learning and 10, 000 test images. Model accuracy was not sacrificed to minimize transmission delay, as shown by the CIFAR-10 dataset utilizing this hierarchical training approach. As a way to decentralize spectrum allocation and utilization, Asad and Bell [22] presents the "Win or Learn Fast (WoLF) custom rate of training." It is based on this premise that a participating device should speed up learning while losing and slow learning down when winning.

The BSs operate FL among numerous mobile users within the SBSs' radio coverage, according to a hierarchical FL framework. For a network-wide agreement, the SBSs send model renewal to a macro-BS frequently. Learning rate is divided into "win" and "loss" when the WoLF agent is successful and unsuccessful in transmitting files, respectively. Win or lose, the WoLF rule is met, since the gadget is slower to pick up skills when it succeeds, and faster to pick up skills when it fails.

For the purpose of temporarily increasing mobile throughput for event attendees, a wireless system is temporarily incorporate within the domain to highlight the importance of picking an appropriate learning rate quantitatively (e.g., football game). For example, using a training rate of 0.01 other than WoLF training rate of 0.01 significantly reduces the likelihood of file transmission failures and file halt, as shown by the simulation results.

Quality of Service (QoS) Provisioning

In massive scale wireless networks, the diminished coupling between individuals in learning aids speed up the process of consolidation. Consequent to that, Q-learning has been applied to RRM (Radio Resource Management) and the spectrum detection issue where agents or BSs do not require to learn each other's explicit strategy. Smart and unstructured BSs have the ability to respond to local input data (e.g., user dispersion and traffic demand) together with incomplete observations of the ecosystem. Consider, for example, the RRM issue at a BS powered by a renewable energy source (RES). A Markov Decision Process (MDP) describes how BSs choose their policies. MDP state vector is $s: = (si)\, I\, i = 1$, whereby si alludes to the BSs state i (that is its level of battery)

There are many different operations in the set, and each one of those operations is used to decide whether or not the operation is ON. Using an agent-dependent incentive (ri), system throughput and battery level may be captured and used to adjust the Q value nearby. Federated Learning (FL) approaches are employed to reduce communication and computation delay while anticipating traffic flow and optimizing user throughput. Wireless edge networks, with lossy and restricted communication networks, are a good fit for FL, as opposed to many centralized learning algorithms that often function in data centers

Because of a limited wireless bandwidth, only a few users are able to coordinate framework or weight renewals at every iteration. The single model updates chosen are those with SINRs that fulfill specific predefined requirements (i.e., channels with a high degree of trustworthiness). It is possible that interference will have an effect on wireless transmissions as well. Zang and Jiang [23] discusses the impact of FL on wireless networks via the use of an algebraic model. Mathematics are used to determine the rate at which FL will converge by taking into consideration both cell coordinations and inter-cell interferences. In a side-by-by-side comparison, the FL convergence speeds of random scheduling, proportional fair and round-robin are all put to the test. Using proportional fair scheduling rather than random scheduling and round-robin has been demonstrated to be more efficient for systems with high SINR goals. As long as the SINR objective is low, round robin has benefits. The FL convergence velocity minimizes fundamentally as SINR targets grow, indicating the necessity to reduce the size of the parameters. A mathematical study reveals a trade-off evident between the sub-channel bandwidth and coordinated users.

Using power-constrained wireless agents and a distant parameter server to develop a shared model, Foukalas [24] evaluated the FL within the wireless system with limited accessibility to Packet Switched (PS) data. The bandwidth-centric fading Multiple Access Channels (MACs) standard is established between participating agents as well as PS to potentially apply DSGD (Decentralized Stochastic Gradient Descent) wirelessly. As a first step, an algorithm that selects one agent for each iteration based on the channel statuses of the devices is proposed. It is entirely up to the coordinated agent to decide whether or not to discretize its gradient estimate (or quantized). The agent sends the quantized bits back to the PS. For transferring analog signals, CA-DSGD utilizes the wireless MAC's additive feature to broadcast the signals. While learning from earlier errors, the agents begin by extracting sparse approximations of their gradients. A lower-dimensional projection of the sparse vector is then created by the devices. From the perspective of the PS, the authors of [25] also provide a system for distributing power that results in consistently aligned gradient vectors. According to simulation results, CA-DSGD converges significantly quicker than D-DSGD and has much effective ccuracy.

Resource Allocation

User association

In [26], an ESN-centric FL approach is analyzed, which determines the location and position of the wirelessly VR (Virtual Reality) clients. Local ESN learning and updating may be performed by many BSs using FL in this approach. All users' positions and orientations are estimated via a collaborative learning algorithm, which has been trained on real-world information.

Power Allocation and Latency

Resource planning and Joint power is deliberated in [27] for (URLLC) Ultra-Reliable Low-Latency Communication with major concentration on automotive system. A concentration on the accurately forecasted queue based on their tail distributions in order to decrease the amount of time spent awaiting in line is one of FL's primary uses. In these tests, there is no concern for the lossy character and bandwidth restrictions of wireless links, which may have an impact on FL quality, throughput, and synchronization.

Bandwidth Allocation and User Selection

In [28], the lack of bandwidth during FL model transmission is explored while FL algorithms are learned over an actual wireless channel. Given that all learning data is sent wirelessly, packet faults and fluctuating wireless channel capacity have an impact on the learning process. At any one time, the BS could only select the users’ subset to amount to FL. The FL loss

function is minimized by learning radio allocation of resources and users' pairing. A closed-form formula for the FL's convergence speed and the effect of model parameters has been developed. User selection and radio channel allocation have been identified as influencing the optimal transmit power for each individual. A convex search may be used to get the optimal radio channel allocation and user selection. The recommended hybrid model of FL and communications might limit FL loss function by approximately 10% and 16% correspondingly, in comparison to i) a better user selections with the probabilistic assignment of resources, and ii) the conventional FL with stochastic procedure of assigning resources and user selection. Thus, the results indicate that FL's accuracy and efficiency are enhanced when learning design and resource distribution are taken into account equally.

Wireless edge applications of FL are examined in [29], where power-centric agents that have localized data collectively develop a framework, which is assisted by remote PS. The objective in this case is to reduce tentative loss functions to the minimum possible value. In order to connect the agents to the PS, a shared wireless connection with limited bandwidth is used. Iteratively, a subset of agents are synchronized to send their localized model renewal to PS via orthogonally assigned frequency network. Every agent should reduce the dimension of its framework renewal to potentially satisfy the capability of its channels. The recommended coordinating approach takes into account the connection states as well as the importance of any local model renewal. When compared to methods that focus primarily on one or the other of the two metrics, this one has a more substantial and longer-lasting impact. Selecting a single agent purposed for communciation at every round generates a better result whenever data is i.i.d amongst various agents. In an event non-i.i.d dataset is present, the learning impact is enhanced by coordinating many agents at each round. As a result, more coordinated agents are needed to distribute data in a less diverse and unbiased manner.

Edge cloud Computing

The edge server establishes different wireless edge agents in order to train ML framework centered on their locally obtained samples of data. A more decentralized form of training improves the system's energy efficiency by enhancing communication and processing. Both Time Division Multiple Access (TDM) and Non-Orthogonal Multiple Access (NOMA) may be used to convey the framework metrics to the servers from the agents' side of the equation. To achieve a training accuracy criteria, it is vital to cut down on overall edge agent energy consumption by optimizing and convexifying the transmission speed and power at the agents for more viable transmissions as well as CPU set-ups for local updating. To increase FEEL's energy efficiency, it has been mathematically shown that both communication and processing may be improved simultaneously, rather than only optimizing one or the other. However, the combined optimization requires an estimate for convexification.

VII. CONCLUSION

A complete review of modern Deep Machine Learning (DML) algorithms for wireless networks has been presented in this article. Power control, edge cloud computing, user association, and spectrum management are just a few of the intriguing DML applications that have been considered. DML has been evaluated for its optimality, convergence rate, scalability, computing cost, and communication overhead. Federated Learning (FL) and partitioned training, two of the most popular DML systems, let wireless technologies to learn a linear model with minimal to no data sharing. As a result, data privacy and transmission costs may be properly safeguarded using this method. Widely used in power management, Quality of Service (QoS) delivery and frequency control when combined with Q-learning or Deep Learning (DL) approaches. In contrast, in edge computing environments, partitioned training is most desirable. It is possible that none of the distributed learning strategies described in this study will attain global optimality quickly. Model partitioning is a feature of partitioned learning, whereas FL does not necessitate the use of pre-existing data.

Data Availability

No data were used to support this study.

Conflicts of Interest

The author(s) declare(s) that they have no conflicts of interest

References

[1]. M. Strategy, "Can Intel's New Knights Landing Chip Compete With NVIDIA For Deep Learning?", Forbes, 2022. [Online]. Doi: https://www.forbes.com/sites/moorinsights/2016/06/28/can-intels-new-knights-landing-chip-compete-with-nvidia-for-deep-learning/. [Accessed: 07- May- 2022].

[2]. P. Nandal, "Deep Learning in diverse Computing and Network Applications Student Admission Predictor using Deep Learning", SSRN Electronic Journal, 2020. Doi: 10.2139/ssrn.3562976.

[3]. L. Fan and L. Zhang, "Multi-system fusion based on deep neural network and cloud edge computing and its application in intelligent manufacturing", Neural Computing and Applications, vol. 34, no. 5, pp. 3411-3420, 2021. Doi: 10.1007/s00521-021-05735-y.

[4]. G. Zhu and T. Zhao, "Deep-gKnock: Nonlinear group-feature selection with deep neural networks", Neural Networks, vol. 135, pp. 139-147, 2021. Doi: 10.1016/j.neunet.2020.12.004.

[5]. Q. Ma, K. Liu, Z. Cao, T. Zhu and Y. Liu, "Link Scanner: Faulty Link Detection for Wireless Sensor Networks", IEEE Transactions on Wireless Communications, vol. 14, no. 8, pp. 4428-4438, 2015. Doi: 10.1109/twc.2015.2421353.

[6]. A. Mallick, S. Dhara and S. Rath, "Application of machine learning algorithms for prediction of sinter machine productivity", Machine Learning with Applications, vol. 6, p. 100186, 2021. Doi: 10.1016/j.mlwa.2021.100186.

[7]. K. Benzekki, A. El Fergougui and A. Elbelrhiti Elalaoui, "Software-defined networking (SDN): a survey", Security and Communication Networks, vol. 9, no. 18, pp. 5803-5833, 2016. Doi: 10.1002/sec.1737.

[8]. "IEEE Cloud Computing Call for Papers Connecting Fog and Cloud Computing", IEEE Cloud Computing, vol. 3, no. 4, pp. c2-c2, 2016. Doi: 10.1109/mcc.2016.83.

[9]. C. Phuah et al., "Abstract 49: White Matter Hyperintensity Spatial Pattern Variations reflect distinct Cerebral Small Vessel Disease Pathologies", Stroke, vol. 50, no. 1, 2019. Doi: 10.1161/str.50.suppl_1.49.

[10]. J. Solares, L. Sboui, Z. Rezki and M. Alouini, "Power Minimization of a Wireless Sensor Node Under Different Rate Constraints", IEEE Transactions on Signal Processing, vol. 64, no. 13, pp. 3458-3469, 2016. Doi: 10.1109/tsp.2016.2548991.

[11]. C. Wang, J. Caja and E. Gómez, "Comparison of methods for outlier identification in surface characterization", Measurement, vol. 117, pp. 312-325, 2018. Doi: 10.1016/j.measurement.2017.12.015.

[12]. K. Chaudhari, "Wheel Defect Detection with Advanced Machine Learning Algorithms", SSRN Electronic Journal, 2019. Doi: 10.2139/ssrn.3729047.

[13]. G. Krishna, D. Niranjan and D. .Shireesha, "Research on the Clustering Algorithm of Component based on the Grade Strategy", International Journal of Engineering Research, vol. 3, no. 12, pp. 757-760, 2014. Doi: 10.17950/ijer/v3s12/1211.

[14]. N. Kumar Kamila, S. Dhal and B. Nayak, "Neural Network Enabled WSN Management for Energy Efficient Routing Mechanism", Indian Journal of Science and Technology, vol. 9, no. 26, 2016. Doi: 10.17485/ijst/2016/v9i26/89824.

[15]. J. Zhao, R. Zhang, Z. Zhou, S. Chen, J. Jin and Q. Liu, "A neural architecture search method based on gradient descent for remaining useful life estimation", Neurocomputing, vol. 438, pp. 184-194, 2021. Doi: 10.1016/j.neucom.2021.01.072.

[16]. Y. Sheikh, "Effective Feature Selection for Feature Possessing Group Structure", International Journal Of Engineering And Computer Science, 2017. Doi: 10.18535/ijecs/v6i5.19.

[17]. Z. Khodkar and J. Abouei, "Energy efficiency enhancement of cell-free massive multiple-input multiple-output network employing threshold-based beamforming", Transactions on Emerging Telecommunications Technologies, vol. 31, no. 7, 2020. Doi: 10.1002/ett.4007.

[18]. E. Ovtchinnikov, "Computing several eigenpairs of Hermitian problems by conjugate gradient iterations", Journal of Computational Physics, vol. 227, no. 22, pp. 9477-9497, 2008. Doi: 10.1016/j.jcp.2008.06.038.

[19]. H. Fang and Q. Qian, "Privacy Preserving Machine Learning with Homomorphic Encryption and Federated Learning", Future Internet, vol. 13, no. 4, p. 94, 2021. Doi: 10.3390/fi13040094.

[20]. L. DE CESARE and A. DI LIDDO, "A STACKELBERG GAME OF INNOVATION DIFFUSION: PRICING, ADVERTISING AND SUBSIDY STRATEGIES", International Game Theory Review, vol. 03, no. 04, pp. 325-339, 2001. Doi: 10.1142/s0219198901000476.

[21]. A. Sarigiannidis and P. Nicopolitidis, "Addressing the interdependence in providing fair and efficient bandwidth distribution in hybrid optical-wireless networks", International Journal of Communication Systems, vol. 29, no. 10, pp. 1658-1682, 2015. Doi: 10.1002/dac.3080.

[22]. A. Asad and M. Bell, "Winning to Learn, Learning to Win: Evaluative Frames and Practices in Urban Debate", Qualitative Sociology, vol. 37, no. 1, pp. 1-26, 2014. Doi: 10.1007/s11133-013-9269-1.

[23]. W. Zhang and S. Jiang, "Effect of Node Mobility on MU-MIMO Transmissions in Mobile Ad Hoc Networks", Wireless Communications and Mobile Computing, vol. 2021, pp. 1-9, 2021. Doi: 10.1155/2021/9954940.

[24]. F. Foukalas, "Federated-Learning-Driven Radio Access Networks", IEEE Wireless Communications, pp. 1-8, 2022. Doi: 10.1109/mwc.102.2100113.

[25]. J. TANG and H. HUANG, "Three dimensional localization algorithm for wireless sensor networks based on projection and grid scan", Journal of Computer Applications, vol. 33, no. 9, pp. 2470-2473, 2013. Doi: 10.3724/sp.j.1087.2013.02470.

[26]. M. Savi and F. Olivadese, "Short-Term Energy Consumption Forecasting at the Edge: A Federated Learning Approach", IEEE Access, vol. 9, pp. 95949-95969, 2021. Doi: 10.1109/access.2021.3094089.

[27]. M. Lezzar and M. Mehmet-Ali, "Optimization of ultra-reliable low-latency communication systems", Computer Networks, vol. 197, p. 108332, 2021. Doi: 10.1016/j.comnet.2021.108332.

[28]. J. Jiang, L. Hu, C. Hu, J. Liu and Z. Wang, "BACombo—Bandwidth-Aware Decentralized Federated Learning", Electronics, vol. 9, no. 3, p. 440, 2020. Doi: 10.3390/electronics9030440.

[29]. J. Qi, Q. Zhou, L. Lei and K. Zheng, "Federated reinforcement learning: techniques, applications, and open challenges", Intelligence & Robotics, 2021. Doi: 10.20517/ir.2021.02.