Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 184 (2021) 419–427

Procedia Computer Science

www.elsevier.com/locate/procedia

The 12th International Conference on Ambient Systems, Networks and Technologies (ANT)
March 23 - 26, 2021, Warsaw, Poland

Federated Learning for Distributed Reasoning on Edge Computing

Ramin Firouzi [a,*], Rahim Rahmani [a], Theo Kanter [a]

[a] *Department of computer and systems sciences, Stockholm University, Stockholm, Sweden*

Abstract

The development of the Internet of Things over the last decade has led to large amounts of data being generated at the network edge. This highlights the importance of local data processing and reasoning. Machine learning is most commonly used to automate tasks and perform complex data processing and reasoning. Collecting such data in a centralized location has become increasingly problematic in recent years due to network bandwidth and data privacy concerns. The easy-to-change behavior of edge infrastructure enabled by software-defined networking (SDN) allows IoT data to be gathered on edge servers and gateways, where federated learning (FL) can be performed: creating a centralized model without uploading data to the cloud. In this paper, we analyze the use of edge computing and federated learning, a decentralized machine learning methodology that increases the amount and variety of data used to train deep learning models. To the best of our knowledge, this paper reports the first use of federated learning to help the Microgrid Energy Management System (EMS) predict load and obtain promising results. Simulations were performed using TensorFlow Federated with data from a modified version of the Dataport site.

Keywords: Distributed Reasoning; SDN; Federated Learning; Edge Computing; Internet of Things; LSTM; Smart Grid.

1. Introduction

The rapid development of the Internet of Things (IoT) led to an exponential growth in network edge data. Performing reasoning and data mining over the data collected from edge devices is of very high interest [1]. Collecting

* Corresponding author.
E-mail address: ramin@dsv.su.se

1877-0509
10.1016/j.procs.2021.03.053

such data at a centralized location has become increasingly problematic in recent years due to network bandwidth and data privacy concerns. Researchers are recently trying to solve these challenges by looking at edge computing since it is closer to the end devices (e.g., smart devices and sensors) than the cloud [2]. Edge computing means to enable the technologies that allow computing to be done at the edge of the network, close to the end devices. Processing data close to the end devices will help solve latency, security, privacy, availability, and power consumption. Edge computing has the advantages of fast processing, energy efficiency, security, mobility, and heterogeneity. Research organizations predict that over 90% of data is stored and processed locally [3]. Federated learning was developed by Google to address this challenge [4]. This method is very close to the well-known server architecture for distributed learning [5], where worker nodes store the raw data. The parameter server maintains the current model and periodically distributes it to the workers, which compute a gradient update and send it back to the server. The server then applies all updates to the central model. This is repeated until the model converges. In federated learning, this framework is optimized to minimize communication between the server and the workers. For this reason, local update computation is more thorough, and compression techniques can be applied when uploading updates to the server. Software-defined networking (SDN) has the potential to alleviate some of the distributed ML problems. An edge network architecture typically includes IoT devices connected to IoT gateways. Gateways and similar devices often have much more compute and storage capacity than IoT devices and are located at the edge of the network, closer to user devices, which means that data does not need to be collected centrally. SDN could be used to easily change the behavior of IoT gateways: Collect data from a group of local IoT devices and perform it in a distributed manner ML or deliver that data securely to nearby edge servers.

This paper's primary purpose is to evaluate the use of edge computing, together with the Federated Learning approach in the STLF challenge for Microgrid Energy Management systems (EMS). Edge computing refers to data processing at the edge of a network instead of cloud or remote server processing. We use Long-short Term Memory (LSTM), a deep neural network for forecasting time series, which uses previous observations of the microgrid electrical load to predict future ones. Our contributions in this work can be summarized as follows: (1) We propose an enabling architecture for FL using edge equipment in the smart grid; (2) We evaluate the potential gain of FL in terms of accuracy through simulations.

2. Related Work

2.1. Reasoning in IoT

Reasoning about an event and inference is essential for the development of IoT applications [6]. From the sensing and processing perspective, the sensing devices monitor a specific area and deliver the captured data to a backend system for processing, event inference, and decision making [7]. Analysis of architectural solutions and case studies on event reasoning and inference mechanisms is discussed in [8]. The backend system in [9] adopts aggregate methodologies for event inference, while in [10] it supports IoT applications for air quality monitoring in indoor environments. Such a system collects contextual data from temperature, humidity, light, and air quality sensors and then centrally infer events. From the quality of inference perspective, event inference utilizing the principles of approximate reasoning like fuzzy logic is proved a useful technique for delivering high-quality inference. The fuzzy logic-based context reasoning model in [11] estimates the radiation levels in the air. The adoption of fuzzy logic aims to handle missing values and, thus, deriving a mechanism capable of delivering alerts. The FL-based fusion model in [12] reduces uncertainty and false-positives within the process of fault detection. In [13], a specific fuzzy logic-based inference system is proposed for ambient intelligence environments. Such a system learns the users' behavior in light of being adapted to the users' profiles. In [14] authors proposed federated reasoning using a type-2 fuzzy engine which locally concludes on an event. Then, through a proposed knowledge-centric nodes clustering scheme in a federated way, nodes disseminate only pieces of inferred knowledge among them to reason unanimously about an event based on their local view. All these works discussed in this subsection are based on rule-based systems. Rules-based systems are deterministic in nature. Rule-based systems can be quite simple but can become rather unwieldy over time as more and more exceptions and rule changes are added. While non-experts may understand individual rules, a full rules engine's complex interaction may be challenging to grasp. Machine learning is an alternative approach that can help to address some of the issues with rules-based methods. Rather than attempt to fully emulate an expert or best

practice's decision process, machine learning methods typically only take the experts' outcomes. In the next subsection, we investigate a few works about machine learning in IoT.

2.2. Internet of Things and Machine Learning

Several papers have been published by integrating deep learning and IoT. One seminal work by Liu *et al.* [15] created a system for identifying foods, in which image preprocessing and segmentation was performed on edge devices, and classification was done on a central server, reducing the latency at inference time. Li *et al.* [16] put forward a similar idea where an ML model was trained in the cloud, then initial layers of the model were distributed to edge servers. At inference time, the edge servers compute the first layer before sending the result of these to the cloud for the final inference, resulting in fewer data sent to the cloud over sending the raw input. Most current work exploring ML and the IoT focuses on centralized training, using IoT/edge computing for inference, or decreasing the size of DNNs for these devices. This article, however, investigates decentralized training using full DNNs.

2.3. Federated Learning

McMahan *et al.* [17] proposed the original FedAvg algorithm to train an aggregate model without uploading client data to a server. FedAvg drastically reduces total communicated data compared to datacenter-style distributed SGD (where each worker performs a single step of SGD before aggregation). FL considers clients with non-IID data distributions, which is an obstacle to training an excellent central model. Zhao *et al.* [18] proposed sharing a small amount of data between non-IID clients to reduce the difference between client distributions, increasing the maximum accuracy their FedAvg models were able to attain. Devices participating in FL are assumed to have highly heterogeneous computing and networking resources. Wang *et al.* [19] addressed this with a control algorithm that dynamically changes the number of local SGD updates that clients perform before uploading their models, based on a tradeoff between the computing power and networking bandwidth available to the client. Several papers have been written on the task of reducing the amount of data communicated during FedAvg. Konecny *et al.* [20] introduced the concepts of *structured* and *sketched* model updates and significantly reduced the quantity of uploaded data during training. However, in our system we used the extreme studentized deviate test before computing the FedAvg algorithm to detect the outliers' weights and doesn't consider them in FedAvg algorithm. This method reduces the amount of data communicated during FedAvg.

3. Motivation and Background

In this paper, we go far beyond the results of our previous work in [21]–[23] by extending the model to federated learning and achieving network edge reasoning via a machine learning system on contextualized data from distributed sources. An IoT edge controller [22], similar to an IoT gateway, can help edge computing by leveraging resource-constrained devices such as the Raspberry Pi. In this regard, in one of our previous papers [21], we proposed a two-level intelligence architecture to provide low-level intelligence closer to the devices that reduce latency and optimize network bandwidth usage and offloads some of the load from the cloud to the edge using fuzzy reasoner. In the other [21], we have used a fuzzy abductive reasoner. We note that the approach in [21], [22] proved to be quite useful for raw data analysis at the edge. However, they did not provide an expression for learning model parameters from data distributed across multiple edge nodes. In this paper, we consider the problem of learning model parameters from data distributed across multiple edge nodes without sending the raw data to a central location. We analyze the convergence bound of distributed gradient descent from a theoretical point of view. We propose a control algorithm that determines the best tradeoff between local update and global parameter aggregation to minimize the loss function under a given resource budget.

4. The Proposed System Architecture

This section describes the proposed architecture of the federated learning system and the overall procedure that allows multiple gateways to control their own IoT devices. The proposed system consists of a cluster head node and N nodes (i.e., workers). The nodes have their own independent environment (i.e., IoT device) and train models with their own data to optimally control the environment through their algorithm. On the other hand, the cluster head node mediates the federated work among the N nodes and ensures that each node's learning process is synchronized.

4.1. Federated learning

The overall flow of the system for the proposed federated learning scheme is shown in figure 1. The process where each worker sends and receives messages back to the cluster head (master node) is called round. For a round, each worker starts a training process using the local data and afterward sends the model gradient to the master node.

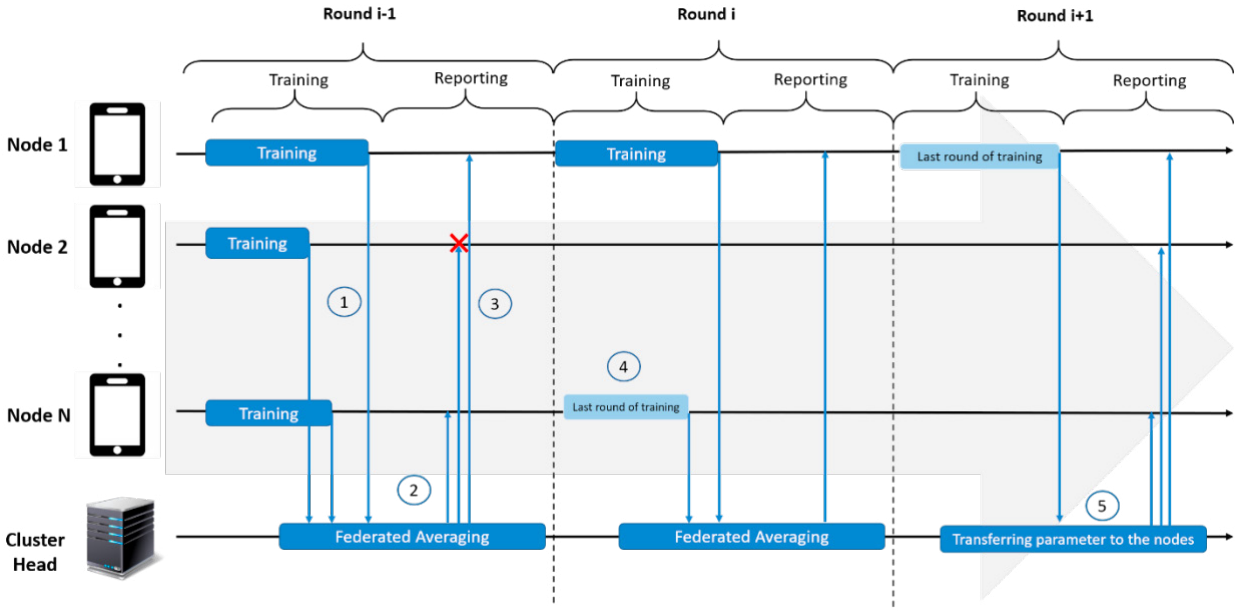

Figure 1. The federated learning overall procedure

In order to optimize the deep learning model, each node of the federated learning algorithm computes the gradients in each round using the local data. The gradients are the vectors of partial derivatives with respect to the models' parameters and are used to find the optimal models to control the IoT device. The role of the master node is to coordinate and mediate gradient sharing model transfer. In the gradient sharing phase, the master node receives the worker nodes' gradient model after each training process (Step 1), as shown in Figure 1, removes the outliers (described in the next subsection), then computes an average of the remaining worker nodes gradients (Step 2), and sends the result back to the workers (Step 3). In the next round's training process, the workers use the average gradient obtained from the master node to optimize the model. The principle of gradient sharing was first developed by [24] for the parallel or distributed SGD implementation that was executed in an asynchronous manner. The predominant disadvantage of the asynchronous method is the problem of delayed gradients, which means that there is noise due to outdated gradients, although it can minimize the time of the worker nodes [25]. The master node does not compute the average until it has received all the gradients from the worker nodes given synchronous gradient sharing. After several rounds of gradient sharing, the worker reaches the predefined convergence criterion to complete the learning process. The last round is the model transfer phase. In this phase, all worker nodes have completed their learning process and send their mature model parameters to the master node (step 4 in Figure 1). The master node considers the mature model parameters it receives from worker nodes as the best ones and transmits them to the rest of all worker nodes (Step 5). after that, all workers replace their model parameters with the mature model parameters.

4.2. Algorithms for Decentralized Distributed Reasoning

In this section, we describe the details of the algorithms used in the cluster head node (master node) and worker nodes. Initially, the deep learning global model is initialized randomly or pre-trained with publicly available data. and this initialized model is the same for all worker nodes since the conditions must be the same for all worker nodes and

only the data used for training is different in each node. In each round, the server sends a global model to each worker nodes. Afterward, worker nodes perform a training epoch, compute new weights, and send them to the master node (see Algorithm 2)—each node then uses SGD to compute the average gradient. In this phase, the master node takes all the weights from the worker nodes and, according to Algorithm 1, based on the Extreme Studentized Deviate (ESD) test, determines the outliers' weights and does not consider them in the calculations of this round. The master node then uses the FederatedAveraging algorithm [26] to calculate the average of the remaining weights and sends them to the worker nodes for the next round. This process is repeated until the model converges.
The centralized model does not match all node data; therefore, personalization is a proposed solution to this problem. Personalization is at the heart of many applications that involve understanding and matching node behavior to it. It consists of retraining the centralized model to create a customized model for each node using node-specific data. This can be done by retraining the model locally for a limited number of epochs using only the node data [27].

4.2.1. Extreme Studentized Deviate (ESD) Test

The extreme studentized deviate test (i.e., Grubbs' Test) is used to detect one or more outliers in a dataset that follows an approximate normal distribution. In order to test that there is an outlier in a dataset, first, we calculate the $G_{calculated}$ and G_{crit}. If $G_{calculated} > G_{crit}$, there is an outlier in the data set, which is the Yi that met the first formula. This process is repeated until $G_{calculated} < G_{crit}$.

$$G_{calculated} = \frac{\max|Y_i - \bar{Y}|}{s} \quad (1) \qquad G_{crit} = \frac{(n-1)t_{crit}}{\sqrt{n(n-2+t_{crit}^2)}} \quad (2)$$

In the first equation $\bar{Y}$ and s denoting the sample mean and sample standard deviation, respectively. In the second equation, t_{crit} is the critical value of the t distribution T(n−2), and the significance level is α/n (α=0.01).

Algorithm 1: Federated Learning (Cluster's head)

for i = 1, 2, 3, ..., M **do**
  P = [];
  for $w \in W$ **do**
    Receive a message m_w from the worker w;
    Append m_w into P
  end
  GCA←Compute $G_{calculated}$; Defined Equation (1)
  GCC←Calculate G_{crit}; Defined Equation (2)
  While (GCA > GCC) **do**
    Remove Y_i;
    GCA←Compute $G_{calculated}$;
    GCC←Calculate G_{crit};
  end
  $w_{t+1} \leftarrow \sum_{k=1}^{M} \frac{1}{M} w_{t+1}^{i}$ The server computes a new global model using this equation;
  Send w_{t+1} to the workers;
end

Table 1. Notation of FL algorithm.

N	N Total number of clients
M	M Clients per round
T	T Total communication rounds
K	K Local steps per round.

Algorithm 2: Federated Learning (Worker w)

1. **for** i = 1, 2, 3, ..., M **do**
2. worker receives model w_r;
3. worker computes average gradient g_k with SGD;
4. worker updates local model;
5. $w_{r+1}^k \leftarrow w_r^k - \eta g_k$;
6. worker sends updated model to server;
7. **end**

5. Performance Evaluation of the System Approach

In this paper, we propose a federated learning system as a practical paradigm to represent the overall control system of a microgrid, the power flow definition, namely the Energy Management System (EMS) as shown in figure 2. The EMS supervises the primary control, also known as local control or internal control, which exclusively relies on local measurements and requires no communication.

5.1. Dataset and Evaluation Method

In this research, we used a modified version of data from Pecan Street Inc's Dataport site. Dataport data set contains circuit-level electricity use data at one-minute to one-second intervals for approximately 800 homes in the United States, with Photovoltaics generation and Electrical Vehicles charging data for a subset of these homes [28]. The dataset is composed of records between January 1st, 2019, and March 31st, 2019, with one-hour resolution data. For this experiment, we selected a subset of 400 clients (Texas) with similar properties from this dataset and added three categories to it: industrial consumption, Emergency facility consumption and renewable energy resource. The data of each client is prepared to be ready for further analysis. First, we transform the data to be on a scale between 0 and 1. Then we transform the time series into sliding windows with look-backs of size 12 and a look-ahead of size 1. Finally, we split data into train and test subsets (90% for training and 10% for the test). We use Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) to evaluate the model's performance concerning the prediction error. RMSE allows us to quantify the error in terms of energy, while MAPE is a percentage quantifying the size of the error relative to the real value. The expressions of RMSE and MAPE are as follows:

$$\mathrm{RMSE}=\sqrt{\frac{\sum_{i=1}^{P}(y_i-\hat{y}_i)^2}{N}} \quad (3) \qquad MAPE = \frac{100\%}{P}\sum_{i=1}^{P}\left|\frac{y_i-\hat{y}_i}{s}\right| \quad (4)$$

where $\hat{y}_i$ is the predicted value, y_i is the actual value, and P is the number of predicted values.

5.2. Simulations Setup

The simulations were conducted with seven nodes. A laptop with a 1.9 GHz Intel core i7 processor and 16GB of memory and Intel UHD Graphics 620 graphic card, as cluster head and ten Raspberry Pi 4 model B as nodes (EMS). We used TensorFlow Federated 0.4.0 with TensorFlow 1.13.1 backend.

Hyper-parameter tuning in deep learning models is vital to obtain the best forecasting performance. However, in this work, we only focus on evaluating the federated learning paradigm. Previous work shows performance insensitivity to combinations of some layers and layer size, as long as we use multiple layers and that the number of hidden nodes is sufficiently large [29]. It was also suggested that very deep networks are prone to under-fitting and vanishing gradients. Following these rules, the initial model hyperparameters (e.g., number of layers and time steps to be considered) were chosen by random search on a randomly selected client's data. The retained model has two LSTM hidden layers composed of 200 neurons each. The loss function used is Mean squared error, and the chosen optimizer is Adam. The model converges around the 20th epoch, and thus, we use close values for rounds and epochs.

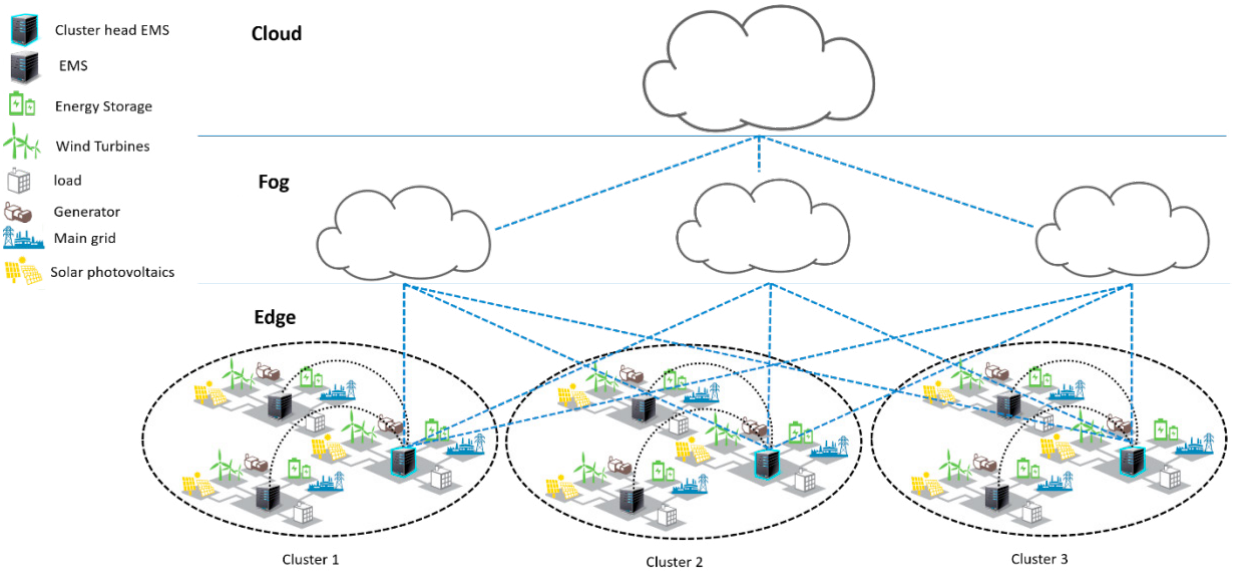

Figure 2. The Scenario of the smart grid

5.3. Numerical Results

5.3.1. Evaluated scenarios

To evaluate the performance of proposed system, different distribution of data among worker nodes were considered as summarized in Table I. The experiment was done three time and each time the number of worker nodes is different (4, 8, 10) to see the effect of larger subsets. The federated learning algorithm was executed for 20 rounds.

Table 2. distribution of data among nodes.

Worker node	Residential Clients pertain to an EMS	Industry clients pertain to an EMS	Emergency-facility clients pertain to an EMS
1	20	1	1
2	50	1	2
3	80	4	2
4	50	4	2
5	80	6	3
6	150	10	5
7	60	8	3
8	100	9	5
9	40	3	5
10	70	4	4

5.3.2. Results for global models

The evaluated scenarios resulted in global models that are obtained following the federated learning approach. These models are evaluated in terms of RMSE and MAPE. Table 3 summarizes the results for the different scenarios. the values of MAPE achieved for various models are reasonable. One of the most notable things we notice is that the global model fits some nodes better than others when considering the fact that not all EMSs have similar profiles. We

also notice that selecting a bigger number of EMSs in each round is preferable, but in cases where sending updates is more expensive in terms of networking, the difference can be compensated by using more local training epochs.

Table 3. Results of RMSE and MAPE for global models.

	RMSE			MAPE		
Number of worker nodes	Min	Max	Mean	Min	Max	Mean
4	0.039	2.63	0.562	8.59%	89.54%	37.21%
8	0.0530	2.624	0.596	11.61%	93.26%	37.98%
10	0.039	2.568	0.559	10.26%	95.94%	38,26%

5.3.3. The behavior of personalization:

In this section, we study the effect of personalization on the performance of the models. First, we test if retraining the model locally for the participant clients gives better results. Then we apply the same thing to the set of clients who did not participate in the training. The models were retrained for 4 epochs for each client. Results for the set of clients participating in the training are summarized in Table 4. We notice an overall improvement of most of the models. For example, subset 1 has an overall improvement of 2.85% in terms of MAPE. However, for some clients, that were treated as outliers, the performance cannot be improved despite retraining, and this is related to the quality of historical data points. Applying the models to these clients' consumption profiles results in very high MAPE, which affects the average results.

Table 4. Results of RMSE and MAPE after personalization.

	RMSE			MAPE		
Number of worker nodes	Min	Max	Mean	Min	Max	Mean
4	0.0	2.38	0.536	7.53%	97.76%	34.36%
8	0.0	2.49	0.551	7.91%	99.16%	36.29%
10	0.0	2.49	0.550	8.24%	98.23%	36.69%

We conclude that we can indeed train powerful models for an EMS using only a subset of the clients forming it. The model can be retrained for applications with high accuracy requirements, resulting in a personalized model that follows the profile's curves better, yielding more accurate predictions.

6. Conclusion

Federated learning enables performing distributed machine learning at the network edge using data from IoT devices. In this paper, we propose a system that leverages edge computing and federated learning to address the data diversity challenges associated with short-term load forecasting in the smart grid. The federated learning in the proposed system, unlike centralized methods, uses edge devices to train models, which reduces the network load. In order to evaluate the performance of both centralized and personalized models in federated settings, we have conducted experiments. The simulation results show a promising approach to building high-performance models with a significantly reduced network load compared to a centralized method. Although federated learning enables machine learning on the device, it suffers from the challenges of security robustness and resource optimization (both computational power and communication). To truly benefit from federated learning in IoT networks, these challenges must be resolved. Future work in this area could explore the development of protocols for federated learning based on device-to-device (D2D) communication.

REFERENCES

[1] S. Wang et al., "When Edge Meets Learning: Adaptive Control for Resource-Constrained Distributed Machine Learning," in IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, Apr. 2018, pp. 63–71, doi: 10.1109/INFOCOM.2018.8486403.

[2] M. Chiang and T. Zhang, "Fog and IoT: An Overview of Research Opportunities," IEEE Internet of Things Journal, vol. 3, no. 6, pp. 854–864, Dec. 2016, doi: 10.1109/JIOT.2016.2584538.

[3] B. R. Kelly and 04/15/15, "Internet of Things Data To Top 1.6 Zettabytes by 2020 -," Campus Technology. https://campustechnology.com/articles/2015/04/15/internet-of-things-data-to-top-1-6-zettabytes-by-2020.aspx (accessed Feb. 04, 2021).

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," IEEE Communications Surveys Tutorials, vol. 19, no. 4, pp. 2322–2358, Fourthquarter 2017, doi: 10.1109/COMST.2017.2745201.

[5] S. Shalev-Shwartz and S. Ben-David, Understanding machine learning: from theory to algorithms. New York, NY, USA: Cambridge University Press, 2014.

[6] C. Gouveia and A. Fonseca, "New approaches to environmental monitoring: the use of ICT to explore volunteered geographic information," GeoJournal, vol. 72, no. 3/4, pp. 185–197, 2008.

[7] A. R. Jaladi, K. Khithani, P. Pawar, K. Malvi, and G. Sahoo, "Environmental Monitoring Using Wireless Sensor Networks(WSN) based on IOT.," vol. 04, no. 01, p. 8.

[8] [8] Palma, D., et al. "Distributed monitoring systems for agriculture based on wireless sensor network technology." International Journal on Advances in Networks and Services 3.1 (2010).

[9] J. Lozano, J. Suárez, P. Arroyo, J. Ordiales, and F. Álvarez, Wireless Sensor Network For Indoor Air Quality Monitoring, vol. 30. 2012, p. 324.

[10] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, New York, NY, USA, Sep. 2002, pp. 88–97, doi: 10.1145/570738.570751.

[11] A.-B. Ramadan, A. El-Garhy, F. Zaky, and M. Hefnawi, "New Environmental Prediction Model Using Fuzzy logic and Neural Networks," International Journal of Computer Science, vol. 9, no. 2, p. 5, 2012.

[12] J. Shell, S. Coupland, and E. Goodyer, "Fuzzy data fusion for fault detection in Wireless Sensor Networks," in 2010 UK Workshop on Computational Intelligence (UKCI), Sep. 2010, pp. 1–6, doi: 10.1109/UKCI.2010.5625598.

[13] H. Hagras, F. Doctor, V. Callaghan, and A. Lopez, "An Incremental Adaptive Life Long Learning Approach for Type-2 Fuzzy Embedded Agents in Ambient Intelligent Environments," IEEE Transactions on Fuzzy Systems, vol. 15, no. 1, pp. 41–55, Feb. 2007, doi: 10.1109/TFUZZ.2006.889758.

[14] C. Anagnostopoulos and K. Kolomvatsos, "Predictive intelligence to the edge through approximate collaborative context reasoning," Appl Intell, vol. 48, no. 4, pp. 966–991, Apr. 2018, doi: 10.1007/s10489-017-1032-y.

[15] C. Liu et al., "A New Deep Learning-Based Food Recognition System for Dietary Assessment on An Edge Computing Service Infrastructure," IEEE Transactions on Services Computing, vol. 11, no. 2, pp. 249–261, Mar. 2018, doi: 10.1109/TSC.2017.2662008.

[16] H. Li, K. Ota, and M. Dong, "Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing," IEEE Network, vol. 32, no. 1, pp. 96–101, Jan. 2018, doi: 10.1109/MNET.2018.1700202.

[17] "Federated learning of deep networks using model averaging."

[18] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated Learning with Non-IID Data," Jun. 2018, Accessed: Jan. 05, 2021. [Online]. Available: http://arxiv.org/abs/1806.00582v1.

[19] S. Wang et al., "Adaptive Federated Learning in Resource Constrained Edge Computing Systems," IEEE Journal on Selected Areas in Communications, vol. 37, no. 6, pp. 1205–1221, Jun. 2019, doi: 10.1109/JSAC.2019.2904348.

[20] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated Learning: Strategies for Improving Communication Efficiency," Oct. 2016, Accessed: Jan. 05, 2021. [Online]. Available: http://arxiv.org/abs/1610.05492v2.

[21] R. Firouzi, R. Rahmani, and T. Kanter, "An Autonomic IoT Gateway for Smart Home Using Fuzzy Logic Reasoner," Procedia Computer Science, vol. 177, pp. 102–111, Jan. 2020, doi: 10.1016/j.procs.2020.10.017.

[22] R. Rahmani and R. Firouzi, "Gateway controller with deep sensing: learning to be autonomic in intelligent internet of things," International Journal of Communication Networks and Distributed Systems, vol. 26, no. 1, pp. 1–29, Oct. 2020, doi: 10.1504/IJCNDS.2021.111631.

[23] R. Firouzi, R. Rahmani, and T. Kanter, "Distributed-Reasoning for Task Scheduling through Distributed Internet of Things Controller," Accepted.

[24] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training," Dec. 2017, Accessed: Jan. 05, 2021. [Online]. Available: http://arxiv.org/abs/1712.01887v3.

[25] Q. Pham et al., "A Survey of Multi-Access Edge Computing in 5G and Beyond: Fundamentals, Technology Integration, and State-of-the-Art," IEEE Access, vol. 8, pp. 116974–117017, 2020, doi: 10.1109/ACCESS.2020.3001277.

[26] A. Hard et al., "Federated Learning for Mobile Keyboard Prediction," Nov. 2018, Accessed: Jan. 05, 2021. [Online]. Available: http://arxiv.org/abs/1811.03604v2.

[27] K. Bonawitz et al., "Practical Secure Aggregation for Federated Learning on User-Held Data," Nov. 2016, Accessed: Jan. 05, 2021. [Online]. Available: http://arxiv.org/abs/1611.04482v1.

[28] Pecan street inc, "Dataport." https://dataport.pecanstreet.org/ (accessed Jan. 05, 2021).

[29] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network," IEEE Transactions on Smart Grid, vol. 10, no. 1, pp. 841–851, Jan. 2019, doi: 10.1109/TSG.2017.2753802.