Federated learning framework for mobile edge computing networks

Romano Fantacci, Benedetta Picano

Department of Information Engineering, University of Florence, Florence, Italy
E-mail: benedetta.picano@unifi.it

Abstract: The continuous growth of smart devices needing processing has led to moving storage and computation from cloud to the network edges, giving rise to the edge computing paradigm. Owing to the limited capacity of edge computing nodes, the presence of popular applications in the edge nodes results in significant improvements in users’ satisfaction and service accomplishment. However, the high variability in the content requests makes prediction demand not trivial and, typically, the majority of the classical prediction approaches require the gathering of personal users’ information at a central unit, giving rise to many users’ privacy issues. In this context, federated learning gained attention as a solution to perform learning procedures from data disseminated across multiple users, keeping the sensitive data protected. This study applies federated learning to the demand prediction problem, to accurately forecast the more popular application types in the network. The proposed framework reaches high accuracy levels on the predicted applications demand, aggregating in a global and weighted model the feedback received by users, after their local training. The validity of the proposed approach is verified by performing a virtual machine replica copies and comparison with the alternative forecasting approach based on chaos theory and deep learning.

1 Introduction

Recently, the ever increasing dissemination in our daily life of intelligent devices such as wearable devices, smartphones, smart cards, sensors, and so on, has triggered the proliferation of numerous distributed network devices generating massive quantity of heterogeneous data to be processed and interpreted [1, 2]. Owing to such unprecedented amount of data with exponential growth trend [3] and the typical private nature of these data, sending all the data to a remote cloud becomes impracticable, unnecessary, and full of privacy concerns [2]. Therefore, all of these factors have contributed to the emergence of the new mobile edge computing (MEC) paradigm [4–6], which exploits the advancement in the storage and computation capacity of modern devices for pushing processing and storing procedures locally on the devices themselves. The MEC approach involves the cooperation of edge-nodes with the remote cloud in order to give rise to a computing system able to support large-scale task processing and managing the environment [1]. Within this context, the efficient and effective handling of big data brings out information and statistical features hidden in the datasets, useful for many application areas such as resource planning, system conditions forecasting, classification, and so on. In this regard, machine learning (ML) [7] techniques have gained momentum to properly catch and interpret data behaviour, by providing a wide range of solutions to analyse datasets trend on the cloud site. Despite the dataset characteristics represent an invaluable source of information to be properly exploited, from the other side the manipulation of users’ sensitive data implies significant responsibilities and risks in keeping them in a centralised site such as the cloud [8]. In order to manipulate big data with respect to users’ privacy, the federated learning (FL) [1, 2, 8, 9] approach has emerged as a set of ML techniques to perform statistical and mathematical training models directly on devices. The FL framework involves ML models locally trained at the devices level, hereafter named as clients, and then the aggregation of these results in a central server, e.g. a base station. Generally speaking, FL has countless issues to deal with, among which [10]

- Non-independent identically distributed data. The clients training datasets are different from each other and a given local training dataset is not representative of the population distribution.
- Unbalanced datasets. The amount of local training data is different for each client. This implies different reliability on trained values for different clients, since too short training procedures may occur.
- Large-scale distribution. The clients involved in the FL framework are significantly greater than the number of data locally processed.
- Limited communication. Mobile devices are not always available to train data and often they may be slow or with poor communication conditions.

The limited communication capacity is out of the objective of this study, which considers a hybrid cloud–MEC network scenario whereby the deployment of virtual machine replica copies (VRCs) of most requested applications on network elements (NEs), located in the network edges, close to the edge devices (EDs) has been taken into account. Therefore, this study focuses on a hybrid cloud–MEC architecture and addresses the design and implementation of the FL framework to properly predict the users’ applications demand in order to perform a proper VRCs allocation in terms of applications hit percentage (AHP). The AHP expresses the percentage of hits in finding the application requested by a device on a NE in its proximity.

The main contribution of this study is

- The contextualisation of the FL framework to the VRCs’ deployment problem, exploiting the decentralised training data is a powerful tool to pursue effective results in the VRCs’ allocation problem.
• The detection and application of basic methods to perform decentralised data training without draining the hardware resources of the EDs.
• The extensive numerical simulations and comparison with the chaos theory (CT) approach, performed to validate the remarkable behaviour resulting from applying the proposed approach to the VRCs’ deployment problem.

The rest of the paper is organised as follows. Section 2 proposes an in-depth review of the related literature. In Section 3, we propose the problem statement, and in Section 4, the FL framework is presented. In Section 5, the experimental results are shown. Finally, the conclusions are drawn in Section 6.

2 Related works

ML techniques constitute a wide branch of big data manipulation literature in MEC networks. Subramaniam and Kaur [11] investigated the application of various ML techniques in order to report the impact of different ML methods on the MEC network. Furthermore, Subramaniam and Kaur [11] analysed the effectiveness of the ML algorithms to detect the presence of malicious attacks in a MEC network. Yu et al. [12] proposed a deep supervised learning method aiming at minimising the overall network cost in performing computational offloading. Differently, a MEC blockchain network has been studied in [13], in which an auction solution based on deep learning is formulated to perform edge resource allocation in order to maximise the edge computing service provider profit. More in depth, Luong et al. [13] built a multi-layer neural network based on the optimal auction solution. In [14], a multi-hidden multi-layer convolutional neural network is adopted to perform data authentication in a robust mobile crowd sensing problem, aiming at improving sensing reliability and reducing the overall latency. By taking into account a real-time industrial application environment, Sangalha et al. [15] address the position-based confidentiality problem in MEC systems by exploiting the k-nearest neighbour and the decision tree approaches. Chang et al. [16] examined the main classes of ML solutions to measure the benefits derived from edge-caching mechanisms, especially in terms of user satisfaction and energy efficiency evaluation.

In contrast, distributed ML is adopted in [17–29]. Tuor et al. [17] used a distributed version of the support vector machine method within an internet of things (IoT) context to evaluate system performance implementing the distributed ML. The distributed stochastic variance reduced gradient is applied in [18], in which the authors aim at optimising the number of collection points to perform data analysis, considering a fixed target accuracy, in order to minimise the amount of network traffic spent to send all the data towards the collection points. In [19], the crowd-sensing problem in an edge-computing scenario is treated by proposing a distributed deep learning approach, in order to lower the traffic congestion in the cloud site and balance the traffic. In particular, Li et al. [19] involve the human in the loop methodology to give a hierarchical structure to the crowd-sensing problem, aiming at controlling the whole crowd-sensing process. Furthermore, the distributed Q-learning algorithm is applied in [20], where the minimisation of the users’ outage is performed by users themselves, selecting the most critical cell on which run the minimisation and considering a heterogeneous networks’ context.

Finally, recently, FL has gained momentum and the authors of [2, 8, 21–24] provided the main examples of such a branch of literature. Yang et al. [21] proposed a novel aggregation data framework for the over the air computation, by exploiting the signal superposition property of wireless channels. The aim of the authors of [2, 21] is the maximisation of the number of devices involved in the aggregation process, by minimising the aggregation error. Wang et al. [22] adopted FL in a MEC system, in which the distributed gradient descent method is applied to determine the best trade-off between local updates and global aggregations, taking into account the minimisation of the loss function subjected to some resource constraints. In the same way, Yu et al. [22] consider as a case study the MEC environment, by proposing the application of the hybrid filtering on stacked encoders to predict the fluctuation of files’ popularity in the contents caching problem. Then, McMahan et al. [8] combined the federated averaging algorithm proposed with the stochastic gradient descent algorithm, in order to train data in a distributed fashion avoiding high level of communication costs. Smith et al. [1] address the multi-task learning problem by resorting to the FL framework based on the novel Mocha context aware optimisation algorithm. A block-chained FL architecture is designed in [23], on the basis of which a distributed consensus strategy is provided by analysing the block-chain end-to-end delay. Finally, FL is proposed in [24] to face the optimisation of the transmission and computation costs in a mixed IoT–MEC network, throughout the application of the multiple deep reinforcements learning agents. With regard to the general problem of the data forecasting an artificial neural network strategy is proposed in [25], which focuses on a multilayer perceptron neural network to forecast the lighting occurrences. Xu et al. [26] iteratively applied recurrent neural networks in order to forecast in real-time the taxi demand in the city of New York. Differently, Chandramutasi et al. [27] addressed the electricity consumption forecasting problem by combining both the feed forward neural network and the long short-term memory strategies. Furthermore, Li et al. [28] focused on the urban traffic passengers flows throughout the spatio-temporal extrapolation of the information regarding the analysed samples series by applying convolutional neural networks and a graph representing the traffic data. Then, the bicycle sharing demand in the city of Beijing has been provided in [29], by proposing an improved version of the Xgboost method combined with the sliding window. The VRC placement has been addressed in [8], considering a MEC environment with heterogeneous applications latency requirements, aiming at minimising the hardware resources consumption for the deployment of the VRCs. Li et al. [19] focused on the optimal virtual machine (VM) replica placement problem for the minimisation of the average system response time and the service provision costs in a Cloud-edge computing (Cloud-EC) scenario. Furthermore, the objective of [30] is the minimisation of the system power consumption, considering the VRCs problem to improve the behaviour of the system in the event of failure. Differently, Zhao et al. [31] proposed a placement strategy based on the divide-and-conquer approach to obtain a near optimal solution to offload the mobile data traffic for distributing the replicas on the infrastructure.

Similarly to [2, 8, 21–24], our study proposes the application of the FL by using straightforward methods belonging to the gradient descent algorithms family. This conservative choice is due to the fact that more complex methods may result in prohibitive consumption of the ED resources, which represents a crucial point in the decentralised data training research field. Different from previous literature, this study focuses on the application of the FL framework to the VRCs deployment problem, by exploiting FL to predict the individual ED demand in order to perform proper VRCs planning. Furthermore, for the best of our knowledge, this is the first paper to contextualise FL to the VRCs allocation problem. Finally, the goodness of the proposed approach has been tested by resorting to extensive numerical simulation and by comparison with other predictive disciplines.

3 Problem statement

3.1 Reference scenario

The reference system scenario consists of the cloud network architecture mixed with MEC as depicted in Fig. 1, where the cloud is located in the remote area of the network, and there is a set of NEs $N^c$ situated close to the EDs. The set hereafter represented by the set $D = \{1, \ldots, j, \ldots, m\}$. Each ED requires computation of one and only one task belonging to the set
\[ T_j = \sum_{i \in N} (\tau_{ij} + c_{ij} \tau_{ij} + (1 - x_{ij})(\tau_{ij C} + c_{ij R}) \] (1)

where \( \tau_{ij} \) and \( \tau_{ij C} \) represent the time spent by task \( j \) on NE \( i \) and on the cloud, respectively. It is important to note that both \( \tau_{ij} \) and \( \tau_{ij C} \) are expressed as the sum of the task execution time spent in the CPU of the NE and the cloud, and the queuing time experienced by the task waiting for its execution on these sites. (The CPU queue has been assumed with the first-in-first-out service policy.) Moreover, \( c_{ij} \) represents the transmission cost in sending the task from ED \( j \) to its nearest NE \( i \). Since the transmission time among NEs and between NEs and the cloud has been supposed negligible, we have taken into account only the \( c_{ij R} \) cost. Finally, \( x_{ij} \) is a binary value equals to 1 if the task \( j \) is computed on NE \( i \), 0 otherwise. It is important to highlight that the OCC in (1) strongly depends on the queuing time experienced by task on the designated computation site. In fact, proper deployment of VRCs on the NE network may drastically reduce the OCC task time.

### 3.2 Problem formulation

The main objective of this study is the maximisation of the hit rate in finding the VRCs of the requested applications on the NEs. This metric evaluation is due to the fact that the deployment of VRCs, provided on the basis of the forecast ED applications demand, lowers the tasks of OCC. Therefore, in formal terms, the AHP can be expressed as

\[ P(X) = \frac{H(X)}{m^{100}}, \] (2)

where \( H(X) \) is the function which, given the VRCs allocation matrix \( X \), whose generic element is \( x_{ij} \), associates the corresponding number of hits occurred in finding the VRCs loaded on the NEs.

Furthermore, the main goal of this study is given by

\[
\max_X P(X).
\] (3)

\[ 0 \leq s_i \leq S, \quad \forall i \in N., \] (4)

where \( S \) expresses the maximum number of SRBs available on an NE. Hence, constraint (4) expresses that each NE has a maximum limitation on the number of SRBs. It is important to note that \( P(X) \) depends on the allocation matrix \( X \), hence on the deployment of the VRCs on the NEs.

### 4 FL framework

#### 4.1 Learning problem

Generally speaking, ML aims at learning parameter models on the basis of some training data. In this regard, an ML model is typically characterised by a loss function depending on the data sample \( z \) and a parameter vector \( w \), i.e. \( f_w(z) \), which captures the error introduced by the model in relation to the training data [2].

By assuming the presence of \( m \) EDs, each of which has local data \( \Delta_j, j = 1, \ldots, m \), the collective loss function can be expressed as [2, 8]

\[ F_j(w) = \frac{1}{|\Delta_j|} \sum_{z \in \Delta_j} f_j(z), \] (5)

where \( |\Delta_j| \) indicates the cardinality of \( \Delta_j \), i.e. the number of elements in \( \Delta_j \). Similarly, from (5) follows that the global loss function computed on all the distributed local dataset \( \Delta_j \) is given by [2, 8]

\[ F(w) = \frac{\sum_{j \in [1, \ldots, m]} |\Delta_j| F_j(w)}{\sum_{j \in [1, \ldots, m]} |\Delta_j|}. \] (6)

As well explained in [2], the direct consequence from (5) and (6), is searching \( w^* \) such that

\[ w^* = \arg \min_w F(w). \] (7)

Therefore, as several previous state-of-the-art works [2, 8], in order to optimise (7) with low-computational complexity, the gradient descent method is applied.

#### 4.2 FL scheme

As depicted in Fig. 2, the proposed FL framework is composed by the clients’ side, responsible for the local data training, and the server side, typically a base station, represented by a central server intended for improving the global learning model, through the merging and aggregation of EDs’ updated local models. This approach is based on the interaction process between the clients and server sides, and, during each algorithm iteration round \( u \), the EDs involved in the training procedure is a subset of the whole EDs set, whose number of elements is equal to \( y \).

The algorithm acts as follows:

- In parallel, each ED \( j \) among the \( y \) EDs involved in the training procedure, updates its local parameter vector \( w_j(u) \), implicitly built
on the basis of its local dataset $\Delta_j$, in accordance with the following rule \cite{2}
\[
    w_j(u) = \tilde{w}_j(u - 1) - \alpha \nabla F_j(\tilde{w}_j(u - 1)),
\]
where $\alpha$ is the learning rate and $\tilde{w}_j(u - 1)$ represents the term $w_j(u - 1)$ after global aggregation.

- The server side provides the weighted average as proposed in \cite{8} and expressed by
\[
    w(u) = \frac{\sum_{j \in \{1, \ldots, m\}} |\Delta_j| w_j}{\sum_{j \in \{1, \ldots, m\}} |\Delta_j|}.
\]

Distributed data training performing the algorithm previously presented implies several advantages in terms of clients’ privacy, preservation of the computational resources of EDs, and messages exchange. In fact, the data training provided locally on the clients’ site, allows users to keep protected their sensitive information. In addition, roughly speaking, the uploading of the ED $j$ parameter vector $w_j$ does not expose the client to any sort of privacy issue, since, given $w_j$, to retrieve $\Delta_j$ is not trivial.

Furthermore, for each algorithm iteration round, the involvement of only a part of the EDs set ensures to reduce the message passing between clients and server sides. Finally, last but not least, it is important to highlight that the usage of the gradient descent algorithm performs optimisation without implying excessive resource consumption, by taking into account the EDs perspective.

The pseudo-code corresponding to the clients and server sides is reported in Algorithms 1 and 2 (see Figs. 3 and 4), respectively.

### 4.3 VRCs allocation

As previously detailed, this study uses FL to provide an accurate distributed prediction on the future EDs applications demand by considering the historical EDs application requests mitigated by the correlation with the other EDs involved in the training procedure. Once the prediction has been achieved, the main goal of this study is the exploitation of this information in order to provide a proper VRCs allocation, in terms of the AHP metric.

Given the predicted application demand, practically expressed in terms of predicted application popularity, the VRCs allocation strategy, given the applications popularity vector $p$ sorted in descending order, consists of the following steps:

1. Start with the NEs empty, hence, with all the SRBs available.
2. Deploy one VRC of each application, starting by that most requested in $p$, on all the NEs in the network with a number of available SRBs able to host the considered application.
3. If any NE cannot host the considered application, the VRCs allocation algorithm terminates, otherwise.
4. Consider the next application in $p$ and repeat steps 2–4 until there exists at least one NE able to host the processed application.

A very relevant point is that the proposed VRCs allocation algorithm provides an unbalanced VRCs deployment, according to the different popularity levels of each application. In particular, step 3 guarantees that the outcome VRCs allocation is not a uniform distribution of the applications. In fact, such a point ensures that no VRCs allocation is provided if the most requested application cannot be stored.

Furthermore, as regards the EDs computation strategy, as previously anticipated, each ED $j$ requesting task computation with application $r$, sends a task to its nearest NE, i.e. $i^*$, which acts in the following three ways:

- The NE $i^*$ computes the task if it contains at least one VRC of the application $r$.
- The NE $i^*$ does not contain any application $r$ VRC, but at least one NE, i.e. $v$, has loaded at least one application $r$ VRC. Therefore, $i^*$ forwards the task to the NE $v$.
- Any NE contains one application $r$ VRC and $i^*$ forwards the task to the far cloud.

---

1. **for each** NE involved in learning process **do**
2. update $w_j(u) = w_j(u - 1) - \alpha \nabla F_j(\tilde{w}_j(u - 1))$;
3. return $w_j(u)$ to the central server;
4. **end for**

**Fig. 3** Algorithm 1: client side

1: initialize $w_0$;
2: **for each** NE involved in learning process **in parallel** do
3: Receive and update $w_j(u)$
4: **end for**
5: update global model $w(u) = \frac{\sum_{j \in \{1, \ldots, m\}} |\Delta_j| w_j}{\sum_{j \in \{1, \ldots, m\}} |\Delta_j|}$.

**Fig. 4** Algorithm 2: server side
The pseudo-code corresponding to the VRCs allocation procedure is reported in Algorithm 3 (see Fig. 5), while Algorithm 4 (see Fig. 6) shows the execution flow of the procedure for the task selection computation site.

### 5 Numerical results

In order to extensively test the proposed FL-based framework, we have resorted to numerical simulations in TensorFlow. We have considered a simulation scenario constituted by $N = 6$ NEs, equipped with a CPU frequency equal to 2.4 GHz, and considering a number of uniformly distributed SRBs $s_i \in [50, 300]$. We supposed that each application occupies a number of SRBs uniformly distributed within the interval $[15, 40]$. Differently, the cloud has been modelled by supposing a CPU frequency equals to 4.6 GHz. The application requests have been simulated as in [22, 32, 33], by using MovieLens 1M dataset [34], hereafter referred to as Dataset 1, and MovieLens 100K dataset [34], named in this paper Dataset 2. Each task has been supposed composed by a number of 64 bits format instructions uniformly distributed within the integer interval $[250, 800]$, requiring 8 CPU cycles per instruction. Furthermore, we assumed the connection link between the EDs and the nearest set equals to 100 Mbit/s.

As regards the loss function, we used the mean squared error (MSE) which, for each data $q_{jl}$ in $D_{j,i}$, is defined as

$$\text{MSE} = \frac{1}{M} \sum_{l=1}^{M} (\hat{q}_{jl} - q_{jl})^2,$$

where $M$ represents the number of samples in test data. Then, in order to test the effectiveness of the proposed approach, we compared the accuracy of the predicted values obtained with the CT approach, through the phase space reconstruction method [35, 36], and with the deep learning-based approach proposed in [27]. The corresponding values adopted for the phase space reconstruction procedure are reported in Table 1. The CT approach has been performed by applying the $k$-neighbours method widely discussed in [37].

The higher accuracy levels in the prediction procedure are clearly evident in Figs. 7 and 8, which depict the MSE behaviour by increasing the time prediction horizon. In Figs. 7 and 8, the accuracy metric trend is showed by varying the prediction horizons in order to test the different approaches. In fact, the

| Dataset | Embedding dimension | Time delay |
|---------|---------------------|------------|
| dataset 1 | 6                   | 9          |
| dataset 2 | 5                   | 7          |

**Table 1 Phase space reconstruction**
accuracy analysis has been performed by taking into account prediction horizons greater than that on which the VRCs placement is actually pursued, i.e. one hour. As it is straightforward to note, the MSE values grow by increasing the prediction horizon. This is due to the intrinsic difficulty in predicting the series of long-term behaviour.

Figs. 9 and 10 highlight system performance by considering the AHP by varying the algorithm communication rounds, for different number of EDs involved in the learning process. Both the figures confirm that greater is the number of EDs participating in the learning process, higher is the number of EDs involved in the process means more significant and accurate information on which the VRCs allocation strategy can act. Furthermore, by increasing the number of algorithm rounds, by which models are updated, the AHP reaches higher values, by improving system performance.

Finally, Figs. 11 and 12 make evident the system improvement on mean task OCC reached by involving a high number of clients in the learning process. These results confirm those previously exposed in Figs. 9 and 10 and, similarly, better performance, i.e. low values of mean OCC are obtained by increasing communication rounds to the considered two-sided framework. All these results validate the goodness of the proposed approach for the VRCs allocation problem and highlight the strict correlation between a valuable prediction model and remarkable system performance. Finally, the resulting system performance makes clear the suitability of FL to our problem.

6 Conclusions

This study addressed the VRCs allocation problem in a hybrid cloud–MEC network, aiming at maximising the AHP probability, i.e. the odds of finding a VRC of the application requested by the EDs on a NE in the edge of the network, instead of in the cloud, typically located in the remote area of the network. The problem has been addressed by applying the FL framework with the gradient descent algorithms family to avoid the excessive exploitation of the EDs hardware resource such as battery lifetime or computational components. Finally, the validity of the proposed framework has been shown throughout extensive empirical evaluation of system performance in comparison with the CT-based predictive approach.

7 Acknowledgments

This work was partially supported by the Project ‘GAUCHO-A Green Adaptive Fog Computing and Networking Architecture’ funded by the MIUR Progetti di Ricerca di Rilevante Interesse Nazionale Bando 2015 under grant no. 2015YPXH4W_004.

8 References

[1] Smith, V., Chiang, C., Sanjabi, M., et al.: ‘Federated multi-task learning’, vol. abs/1705.10467, 2017
[2] Wang, S., Tuor, T., Salonidis, T., et al.: ‘When edge meets learning: adaptive control for resource-constrained distributed machine learning’, vol. abs/1804.05271, 2018, pp. 63–71
[3] Chiang, M., Zhang, T.: ‘Fog and IoT: an overview of research opportunities’, IEEE Internet Things J., 2016, 3 (6), pp. 854–864
[4] Shan, X., Zhu, H., Li, P., et al.: ‘A survey on computation offloading for mobile edge computing information’
[5] Mach, P., Becvar, Z.: ‘Mobile edge computing: A survey on architecture and computation offloading’, IEEE Commun. Surv. Tutor., 2017, 19, (3), pp. 1628–1656
[6] Liu, H., Eldarrat, F., Alqahtani, H., et al.: ‘Mobile edge cloud system: architectures, challenges, and approaches’, IEEE Syst. J., 2018, 12, (3), pp. 2495–2508
[7] Zhang, C., Patran, P., Haddadi, H.: ‘Deep learning in mobile and wireless networking: A survey’, vol. abs/1803.04311, 2018
[8] McMahan, H.B., Moore, E., Ramage, D., et al.: ‘Federated learning of deep networks using model averaging’, vol. abs/1602.05629, 2016
[9] Zheng, Q., Liu, Y., Chen, T., et al.: ‘Federated machine learning: concept and applications’, vol. abs/1902.04885, 2019
[10] McMahan, H.B., Moore, E., Ramage, D., et al.: ‘Communication-efficient learning of deep networks from decentralized data’, 2017
[11] Subramaniam, P., Kaur, M.J.: ‘Review of security in mobile edge computing with deep learning’, March 2019, pp. 1–5
[12] Yu, S., Wang, X., Langar, R.: ‘Computation offloading for mobile edge computing: A deep learning approach’ 2017 IEEE 28th Annual Int. Symp. on Personal, Indoor, and Mobile Radio Communications (PIMRC), Montreal, Canada, October 2017, pp. 1–6
[13] Luong, N.C., Xiaoyi, Z., Wang, P., et al.: ‘Optimal auction for edge computing resource management in mobile blockchain networks: a deep learning approach’, May 2018, pp. 1–6
[14] Zhou, Z., Liao, H., Gu, B., et al.: ‘Robust mobile crowd sensing: when deep learning meets edge computing’, IEEE Netw., 2018, 32, (4), pp. 54–60
[15] Sangaiah, A.K., Medhat, D.V., Han, T., et al.: ‘Enforcing position-based confidentiality with machine learning paradigm through mobile edge
computing in real-time industrial informatics’, *IEEE Trans. Ind. Inf.*, 2019, 15, (7), pp. 4189–4196.

[16] Chang, Z., Lei, L., Zhou, Z., et al.: ‘Learn to cache: machine learning for network edge caching in the big data era’, *IEEE Wirel. Commun.*, 2018, 25, (3), pp. 28–35.

[17] Tuer, T., Wang, S., Salonidis, T., et al.: ‘Demo abstract: distributed machine learning at resource-limited edge nodes’, April 2018, pp. 1–2.

[18] Valero, L., Passarella, A., Conti, M.: ‘Optimal trade-off between accuracy and network cost of distributed learning in mobile edge computing: an analytical approach’, June 2017, pp. 1–9.

[19] Li, L., Ota, K., Dong, M.: ‘Human in the loop: distributed deep model for mobile crowdsensing’, *IEEE Internet Things J.*, 2018, 5, (6), pp. 4957–4964.

[20] Kudo, T., Ohtsuki, T.: ‘Cell selection using distributed q-learning in heterogeneous networks’. 2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conf., Kaohsiung, Taiwan, October 2013, pp. 1–6.

[21] Yang, K., Jiang, T., Shi, Y., et al.: ‘Federated learning via over-the-air computation’, December 2018.

[22] Yu, Z., Hu, J., Min, G., et al.: ‘Federated learning based proactive content caching in edge computing’, December 2018.

[23] Kim, H., Park, J., Bennis, M., et al.: ‘Blockchain-based on-device federated learning’, 2019, p. 1.

[24] Ren, J., Wang, H., Hou, T., et al.: ‘Federated learning-based computation offloading optimization in edge computing-supported internet of things’, *IEEE Access*, 2019, 7, pp. 69194–69201.

[25] Abdullah, N.H., Adnan, R., Samad, A.M., et al.: ‘Lightning forecasting modelling using artificial neural network (ANN): case study Sultan Abdul Aziz Shah Airport or Skypark Subang’. 2018 IEEE Conf. on Systems, Process and Control (ICSPC), Malacca, Malaysia, December 2018, pp. 1–4.

[26] Xu, J., Rahmatiazadeh, B., Bölöni, L., et al.: ‘Real-time prediction of taxi demand using recurrent neural networks’, *IEEE Trans. Intell. Transp. Syst.*, 2018, 19, (8), pp. 2572–2581.

[27] Chandramitasari, W., Kurniawan, B., Fujimura, S.: ‘Building deep neural network model for short term electricity consumption fore–casting’. 2018 Int. Conf. on Advanced Intelligent Informatics (SAIN), Yogyakarta, Indonesia, August 2018, pp. 43–48.

[28] Li, J., Peng, H., Liu, L., et al.: ‘Graph CNNs for urban traffic passenger flows prediction’. 2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/IUC/ATC/CBDCom/IOP/SCI), Guangzhou, China, October 2018, pp. 29–36.

[29] Cui, Y., Lv, W., Wang, Q., et al.: ‘Usage demand forecast and quantity recommendation for urban shared bicycles’. 2018 Int. Conf. on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), Zhengzhou, China, October 2018, pp. 238–2388.

[30] Zhao, L., Liu, L.: ‘Optimal placement of virtual machines for supporting multiple applications in mobile edge networks’, *IEEE Trans. Veh. Technol.*, 2018, 67, pp. 6533–6545.

[31] Zhao, L., Liu, J., Shi, Y., et al.: ‘Optimal placement of virtual machines in mobile edge computing’. GLOBECOM 2017–2017 IEEE Global Communications Conf., Singapore, 2017, pp. 1–6.

[32] Müller, S., Atan, O., van der Schara, M., et al.: ‘Context-aware proactive content caching with service differentiation in wireless networks’, *IEEE Trans. Wirel. Commun.*, 2017, 16, (2), pp. 1024–1036.

[33] Li, S., Xu, J., van der Schara, M., et al.: ‘Popularity-driven content caching’. IEEE INFOCOM 2016 – The 35th Annual IEEE Int. Conf. on Computer Communications, San Francisco, California, April 2016, pp. 1–9.

[34] Harper, F.M., Konstan, J.A.: ‘The movielens datasets: history and context’. *ACM Trans. Interact. Intell. Syst.*, 2015, 5, (4), pp. 19:1–19:19.

[35] Liu, Z.: ‘Chaotic time series analysis’, *Math. Probl. Eng.*, 2010, 2010, pp. 1–31.

[36] Takens, F.: ‘Detecting strange attractors in turbulence’, in Rand, D., Young, L.-S. (Eds.): ‘Dynamical systems and turbulence, Warwick 1980’ (Springer, Berlin, Heidelberg, 1981), pp. 366–381.

[37] Kantz, H., Schreiber, T.: ‘Nonlinear time series analysis’ (Cambridge University Press, Cambridge, 2003, 2nd edn.)