Estimation of Individual Device Contributions for Incentivizing Federated Learning

Takayuki Nishio  
Graduate School of Informatics, Kyoto University, Kyoto, Japan  
nishio@i.kyoto-u.ac.jp

Ryoichi Shinkuma  
Graduate School of Informatics, Kyoto University, Kyoto, Japan  
shinkuma@i.kyoto-u.ac.jp

Narayan B. Mandayam  
WINLAB, Rutgers University, NJ, USA  
narayan@winlab.rutgers.edu

Abstract—Federated learning (FL) is an emerging technique used to collaboratively train a machine-learning model using the data and computation resources of mobile devices without exposing private or sensitive user data. Appropriate incentive mechanisms that motivate the data and mobile-device owner to participate in FL is key to building a sustainable platform. However, it is difficult to evaluate the contribution levels of participants to determine appropriate rewards without large computation and communication overhead. This paper proposes a computation- and communication-efficient method of estimating participants contribution levels. The proposed method requires a single FL training process, which significantly reduces overhead. Performance evaluations are done using the MNIST dataset, showing that the proposed method estimates participant contributions accurately with 46–49% less computation overhead and no communication overhead, as compared to a naive estimation method.

Index Terms—Federate Learning, Incentive Mechanism, Contribution Estimation, Contribution Metric

I. INTRODUCTION

The expanded use of machine learning (ML) has empowered a wide variety of internet-of-things (IoT) applications, including fine-grained road-traffic, pedestrian, and environmental prediction with anomaly detection for network systems and fraud detection for financial transactions [1], [2]. Generally, ML requires tremendous computational power to quickly produce analytical results. However, the steady growth of cloud-computing platforms (e.g., Amazon Web services, Microsoft Azure, and Alibaba) has recently enabled sufficient support.

Data used by ML for IoT applications are generated by the IoT devices at the network edge that include sensors, smartphones, appliances, and smart vehicles. People and organizations engaged in IoT activities have identified concerns about the privacy of their data when sharing with IoT third parties. After the European Union adopted its general data protection regulation (GDPR) [3], privacy became a top-prioritized issue in Europe with regard to IoT applications. The regulation also addresses bandwidth costs, which individuals or private organizations must bear to support their IoT needs. As the volume of data increases, such bandwidth costs will become more burdensome [4], [5].

Federated learning (FL) was invented to tackle the above issues [6]. With traditional ML, a training dataset is usually stored at a central entity. Data must be first collected from a variety of sources to facilitate the learning process. FL provides data integration methods that comply with privacy and security laws. Under FL, data owners employ privacy protection techniques (e.g., homomorphic encryption, secret sharing, and differential privacy) to contribute sensitive trained data to a federated source, which then combines the local model parameters to train a more effective collective ML model. This allows the learning process to leverage the computational power of distributed data sources, similar to crowdsourcing.

Although FL has been advantageous for collaborative learning because of its data privacy and protection measures, it still faces an open challenge of incentivizing data owners devices to join the FL effort [7]. One idea is to reward participants according to their contributions. However, accurately evaluating their contributions is quite difficult. Furthermore, it has been reported that the relationship between model accuracy and the amount of training data is nonlinear. Thus, accuracy depends on model complexity and data quality. These measures can hardly be predicted in advance.

Generally, with FL, data are neither balanced nor independently and identically distributed (non-i.i.d) among clients. Thus, as client training data are collected based on distributed environments and usage patterns, both the size and the distribution of the local datasets will vary greatly [8]. Therefore, accurate estimation of individual contribution levels with small computation and communication overhead is key to the success of incentivizing FL participants.

This paper proposes a method that estimates the individual contribution level of FL participants, requiring no overhead communications traffic and little computation overhead. Conventionally, a direct and accurate method of estimating individual contribution levels is to first measure the degradation

This work is supported in part by the US NSF under Grant ACI-1541069, JST PRESTO Grant no. JPMJPR1854, and KDDI foundation. Also, the research results were partly obtained from the commissioned research by National Institute of Information and Communications Technology (NICT), Japan.
of model accuracy by removing the local model provided by each participant. This method is more computation- and communication-resource consuming, because it must repeat entire FL processes as many times as the number of participants. This is helpful for the incentive mechanism of FL when quickly determining rewards for each participant on the basis of their contribution levels. Of course, accuracy of the estimation must be also ensured. In this paper, performance evaluation using a real dataset is conducted to validate the proposed method in terms of estimation accuracy.

II. PRIOR WORKS

Several studies on FL incentive mechanisms were published in 2019 and 2020, suggesting that this research topic is building momentum. Several basic analyses have been presented [9]–[12]. Pandey et al. designed a framework in which FL-participating clients iteratively solved local learning sub-problems to meet an accuracy level that was subject to an offered incentive [9]. A communication-efficient cost model for the participating clients was established to formulate the incentive mechanism and to induce the necessary interaction between the mobile-edge computing (MEC) server and the participating FL clients. Introducing a two-stage Stackelberg game, they analyzed the game’s equilibria and the response behaviors of the participating clients. Lee et al. designed a distributed learning resource-management mechanism over multiple MECs owned by different operators [10]. They also presented a game theoretic approach that focused on analyzing the market behaviors and the economic benefits of FL by formulating and analytically solving a Stackelberg game model. Kang et al. adopted contract theory to design an efficient incentive mechanism that mapped contributed resources into appropriate rewards to entice mobile-device owners possessing high-quality data to join FL to overcome information asymmetry issues [11]. They presented the problem formulation and the optimal contract design of their mechanism and showed its superiority using the Stackelberg game model. Chen et al. modeled and formulated the mechanism-design problem with type-imposed externalities [12]. For quasi-monotone externality-settings, they provided a revenue-optimal and truthful mechanism. For the general valuation functions, they provided both necessary and sufficient conditions for all truthful and individually rational mechanisms.

The method of determining rewards for clients is a key issue for FL incentive mechanisms [13]–[15]. Kang et al. applied reputation as the necessary metric needed to assess the reliability of an FL-worker candidate, thus ensuring reliable worker selection. High-reputation workers bring high-quality data (i.e., high accuracy and reliability) to model training, generating reliable local model updates for FL tasks. Liu et al. suggested that, to properly incentivize data owners to contribute their efforts, the Shapley value (SV) was often adopted to fairly assess their contributions [14]. To help FL systems compute SVs to support sustainable incentive schemes, they proposed a blockchain-based peer-to-peer payment system: FedCoin. The SV of each FL client reflected its contribution to the global FL model in a fair way. It was calculated using the Proof-of-Shapley consensus protocol, which replaced the traditional hash-based protocol in existing blockchain proof-of-works methods. Toyoda and Zhang introduced a competitive model-update method that allowed any rational worker who followed the protocol to maximize their profits. Each worker chosen in a certain round selected top model updates submitted by workers in the previous round and updated their own model. Here, workers’ reward was decided by the vote of the next round of workers. The motivation for choosing this kind of model is that participants model updates will have a greater chance to be up-voted in the next round, meaning that more rewards will be obtained. This design naturally compels rational workers to behave honestly in an environment lacking heavy cryptography or special hardware.

Sustainability and fairness are also key issues of FL incentive mechanisms [7], [16], [17]. Yu et al. considered a method to quantify the payoff for each data owner in order to achieve long-term systemic wellbeing [16], [17]. Participants incurred some costs for contributing to the FL model with their local datasets, and the training and commercialization of the models took time. Thus, delays were incurred between data federation and budget solvency for paying the participants. To address the issues of unshared decisions and ambiguous contribution evaluations, Zhan et al. designed an algorithm based on deep reinforcement learning (DRL), in which models can learn system states from historical training records [7]. DRLs can adjust the strategies of the parameter server and edge nodes according to environmental changes that may impose different requirements on training data.

Incentive mechanisms for FL in heterogeneous resource environments have also been studied [18]–[20].

To the best of our knowledge, no prior work has addressed the simultaneous reduction of computation and communication overhead for estimating the individual contribution levels of FL participants.

III. PROPOSED METHOD

A. System model

Figure 1 shows our system model. The system comprises a server and multiple FL clients. At each client, the sensor generates data to be used for ML. The receiver receives an ML model from the server and stores it. The learner performs the training process to update the ML model using the data obtained from the sensor. The transmitter uploads the updated ML model to the server. The server ultimately receives the updated ML models from all clients and stores them. The aggregator performs model aggregation to update the ML model using the models obtained from the clients. The evaluator estimates the individual contribution levels of each client by comparing the ML models before and after being updated. The evaluator then sends back the rewards...
to each client based on the level of contribution. Note that
decision making by clients is out of the scope of this paper
and will be included in future work. The decision maker at
each client, however, makes the decision to participate in FL if
the reward is sufficient to compensate the costs of contribution.
Otherwise, the client leaves the FL.

B. Design metrics to evaluate client contributions

Evaluating the contributions of each client accurately is
difficult, because explicitly describing the actual improvement
to model performance based on the quality of the clients’
data and model (e.g., data amount, data variety, noisy label,
the number of epochs, and noisy gradients) is complicated.
Therefore, we next discuss model performance-based metrics
that can be used to estimate the contributions of each client.

Let \( C = 1, ..., c \) be a set of indices that describes \( c \) clients.
The amount of data samples possessed by client \( i \) and the
data distribution are denoted as \( d_i \) and \( v_i \), respectively. \( M_i^{(r)} \)
denotes the model updated by client \( i \) at round \( r \), and \( M_S^{(r)} \)
denotes the model aggregated by those updated by a subset
of clients, \( S \subseteq C \). \( M^0 \) is defined as an initial model of
the parameters that are randomly determined. \( P(M) \) is the
performance score of model \( M \), and the validation accuracy
or loss can be used for the score.

In each round, each client updates \( M_i^{(r-1)} \) with its own
data and uploads an updated model, \( M_i^{(r)} \). The server then
aggregates the models into \( M_S^{(r)} \). The round is repeated until
the \( P(M_S^{(r)}) \) or \( r \) achieves a certain threshold.

Naive contribution metric

Let \( M_S \) be a global model at the end of the FL where only
clients in \( S \) participate. We define a naive metric for evaluating
contributions of client \( i \) as the gain in performance when the
client joins the FL. The gain can be determined as the relative
performance of a model that includes all clients but excludes
only client \( i \). Specifically, the normalized relative gain is given as

\[
G_i^{\text{Naive}} = \frac{P(M_C) - P(M_{C\setminus\{i\}})}{\sum_{i \in C} P(M_C) - P(M_{C\setminus\{i\}})},
\]

where \( P(M_{C\setminus\{i\}}) \) indicates a performance score in which all
clients, excluding client \( i \), train the model. The naive metric
is intuitive, reasonable, and calculable. However, calculating
the metric requires additional FL training to obtain \( M_{C\setminus\{i\}} \),
which causes large computational and communication overhead.
Specifically, this overhead of calculating the naive metric
can be given as

\[
\text{Comp.} : \sum_{i \in C} r_{\text{end}} (C_{\text{server}} + \sum_{j \in C \setminus \{i\}} C_{\text{ud}}^{(j)}), \quad (2)
\]

\[
\text{Traffic} : \sum_{i \in C} r_{\text{end}} \sum_{j \in C \setminus \{i\}} 2\Theta_{\text{model}}^{(j)}, \quad (3)
\]

where \( r_{\text{end}} \) is the number of rounds at the end of the FL
task. \( C_{\text{ud}}^{(j)} \) and \( C_{\text{server}} \) denote the amount of computations
for model updating using stochastic gradient decent (SGD)
operations at client \( j \) and the computation at the server for
model aggregation and model validation. \( \Theta_{\text{model}}^{(j)} \) denotes
the traffic required for transmitting a model to or from client \( j \).
The usual FL process causes computations of model aggrega-
tion and model updates at each client, the sum of which is
\( C_{\text{server}} + \sum_{j \in C} C_{\text{ud}}^{(j)} \) for each round. Traffic is also generated
for distributing and gathering models to or from FL clients at
each round. The round is repeated \( r_{\text{end}} \) times. For calculating the
naive metric for client \( i \), the FL process with \( C \setminus \{ i \} \) must
be conducted individually. Thus, the additional computation
and traffic are as shown in (2) and (3). The computation and
communication overhead can be huge, because the number of
clients are expected to be large in mobile networks and the
data size of the model may be huge, especially when using
deep neural networks.

Therefore, we require an intuitive, reasonable, and,
computation-and communication-efficient metric to evaluate
client contributions.

Step-wise contribution

We propose a light-weight but intuitive contribution estimation
method based on a step-wise contribution calculation. The
metric used in the proposed method is defined as the sum of
gains that include the client model in each round, calculated
as

\[
G_i^{\text{SWC}} = \frac{\sum_{r=1}^{r_{\text{end}}} P(M_S^{(r)}) - P(M_{C\setminus\{i\}}^{(r)})}{\sum_{r=1}^{r_{\text{end}}} P(M_C^{(r)}) - P(M_{C\setminus\{i\}}^{(r)})}, \quad (4)
\]

where the denominator is used for normalization. The metric
evaluates how much the client’s model improves the global
model at each round and regards the sum of the step-wise
contributions as the contribution of the FL client. This is based
on the intuition that a client that improves model performance
at each round will also contribute to the improvement of the
final overall model performance.

Note that \( G_i^{\text{SWC}} \) cannot be the same as \( G_i^{\text{Naive}} \), because
\( M_j^{(r)} \), used to obtain \( M_{C\setminus\{i\}}^{(r)} \), is calculated from \( M_C^{(r-1)} \),
which involves the effect of \( M_i^{(r-1)} \). However, \( M_{C\setminus\{i\}}^{(r)} \) never
involves the effect of client \( i \). Thus, the proposed metric can
be calculated in \( r_{\text{end}} \) FL training rounds. That is, the FL
operator does not need the additional FL training rounds,
whereas the naive metric requires \((c + 1) \cdot r_{end}\) FL training rounds. The proposed method requires an increase in model aggregations, model validations, and gain calculations by a factor \(c \cdot r_{end}\), compared with the original FL training process. The computation and traffic overhead caused by the proposed method is described as

\[
\text{Comp.} : \quad r_{end} \cdot c \cdot C_{\text{server}},
\]

\[
\text{Traffic} : \quad 0.
\]

The additional computation of the \((c - 1)\)-times model aggregation is triggered by the server during each round. The calculation is completed by the server, and no traffic overhead is incurred. Compared with the naive metric, the computation overhead is reduced by

\[
\frac{c \cdot C_{\text{server}}}{c \cdot C_{\text{server}} + \sum_{i \in C} \sum_{j \in C \setminus \{i\}} C_{ij}}.
\]

Generally, because the computation power of the server is much higher than those of clients, the additional computation of the proposed method is not a critical issue.

Apart from incentive and reward selection, the proposed metric can also be used for mechanisms such as client selection, where the FL operator selects a subset of clients to participate to reduce the latency of model distribution and to expedite uploading to maintain the improvements to model performance [21].

### Other heuristic metrics

Based on the intuition that clients having copious and diverse data improves the global model, two heuristic metrics can be defined:

\[
\text{Heuristic1} : \quad G_{\text{H1}} = D_i / \sum_{i \in C} D_i,
\]

\[
\text{Heuristic2} : \quad G_{\text{H2}} = D_i \cdot v_i / \sum_{i \in C} D_i \cdot v_i,
\]

where \(D_i\) denotes the number of data sample stored by client \(i\), \(v_i\) denotes the variety of the data owned by client \(i\), which refers specifically to the number of classes of the data samples stored by client \(i\) in classification tasks and the range of the target values in the regression tasks. These metrics are intuitive and easy to calculate. Furthermore, they can be obtained prior to the FL training process. However, the metrics do not work in some cases. Supposing that there are three clients having the same number of data with four classes at each client, the client’s data class differs from those of the others. The other clients’ data classes overlap somewhat. In this case, the heuristic metrics become the same value for each client. However, their actual contributions may differ.

### IV. EXPERIMENTAL EVALUATION

#### A. Setup

A total of \(c = 3\) clients participated in the FL for all rounds. The clients trained their local models using their own data, uploading them to a server that aggregated the models into a single global model. We adopted an image classification task leveraging the MNIST dataset [22], which is a dataset of handwritten digits commonly used as a benchmark. The MNIST dataset consists of 60,000 training images and 10,000 testing images with digits of 0–9 stored as 28 \(\times\) 28 pixels.

The small portions of the training dataset were distributed to \(c = 3\) clients. The total number of data samples stored by Client 3 was fixed to \(D_3 = 350\), and those of Clients 1 and 2 were \(D_1, D_2 \in [50, 200, 350]\). We considered a non-iid setting, wherein each client possessed data samples from specific classes of the 50,000 training samples. Specifically, Client 1 stored \(D_1\) samples by randomly sampling from \(v_1 = 7\) classes of digits (i.e., 0–6). Client 2 stored \(D_2\) samples by randomly sampling from \(v_2 = 7\) or 5 classes of digits (i.e., 0 or 2 to 6). Client 3 stored 350 samples by randomly sampling from \(v_3 = 3\) classes of digits that did not overlap the classes Client 1 (i.e., 7–9). In these settings, Client 1 stored samples of the most various classes \((v_1 > v_2 > v_3)\), and Client 2 stored digits that were stored by Client 1. The digits of 7–9 were stored by only Client 3. Therefore, the contributions of Client 2 were expected to trend smaller than those of the other clients. The 10,000 testing images were used for the model validation at each round.

We implemented convolutional neural networks as global and local models using TensorFlow [23]. Specifically, the model consisted of two \(3 \times 3\) convolution layers of 16 and 32 channels, each of which was activated by a rectified linear unit (ReLU) and batch normalized. Each convolution layer was followed by 2 \(\times\) 2 max pooling. The last max-pool layer was followed by two fully connected layers of 64 units with ReLU activation and another 10 units activated by soft-max. The batch size, the number of epochs for local training \((E)\), and the number of FL rounds were set to \(B = 50\), \(E = 30\), and 30, respectively. An SGD having a learning rate of 0.25 was used for the optimizer of each client.

We evaluated the errors of the metrics defined in Sect. III-B from the naive contribution metric. The error was defined as the Euclidean distance between the metrics, which is written as follows:

\[
E = \sqrt{(G_{\text{H1}} - G_{\text{H1}})^2 + (G_{\text{H2}} - G_{\text{H2}})^2 + (G_{\text{Naive}} - G_{\text{Naive}})^2}.
\]

#### B. Results

Figure 2 depicts a total computation times required for the FL and its evaluating contributions, which did not include latency for model transmissions. Thus, this result indicates the computation load of each method. Because the computation load to calculate heuristic metrics was negligible, which was hundreds microseconds in the experiments, the computation time of the heuristic methods were almost the same as the usual FL. On the other hand, the naive and proposed method increased the computation time. However,
the proposed method increased a much shorter time than did the naive method, meaning that the proposed method required much smaller computation overhead, as discussed in Sect.III-B.

Figure 3 depicts the contribution scores of each client when Clients 1 and 2 had 350 training samples of 0–6 digits, and Client 3 has 350 samples of 7–9 digits. For each method, the contribution scores of Clients 1 and 2 were nearly the same. This is reasonable scoring, because Clients 1 and 2 had the same number and variety of training samples. On one hand, the score of Client 3 was higher than the others according to the naive and proposed methods. This is also reasonable, because the data stored by Client 3 were unique, whereas training data for integers 0–6 were stored at both clients. When Client 3 left the FL, the global model could not achieve an accuracy higher than 0.7. The model could, however, achieve greater than 0.7 if Clients 1 or 2 left, because both had training data of 0–6 digits. The heuristic methods for Clients 1 and 2 did not consider the data uniqueness and gave the same or lower score to Client 3, which is unreasonable in this scenario.

Figure 4 depicts the contribution scores of each client when Clients 1, 2, and 3 had 350 training samples of 0–6 digits, 2–6 digits, and 7–9 digits, respectively. In this setting, the contributions of Client 2 must have been smaller than Client 1, because Client 2 had a smaller variety of training samples. The naive and proposed methods gave the lower score to Client 2 as expected, whereas the heuristic methods did not.

Figure 5 depicts the errors of scores from the naive method when changing the data amount of Clients 1 and 2, $D_1$, $D_2$. Here, Clients 1 and 2 had training samples of 0–6 and 2–6 digits, respectively. The proposed metric achieved a smaller gap from the naive method when the training samples of Clients 1 and 2 became larger, whereas the gaps in the other methods became larger.

Figure 6 shows the errors that occurred when Clients 1 and 2 had the same classes of training samples. This result highlights cases in which the proposed method achieved larger errors than did the other methods. When $D_1$ and $D_2$ were larger than 200, the proposed method achieved lower errors than did other methods, as was the case with the results shown in Figs. 3–5. On the other hand, when $D_1$ or $D_2$ was 50, the error of the proposed method became larger than those of other methods. When $D_1$ or $D_2$ was 50, the proposed method estimated that the contribution of the client having 50 data samples was less than other clients but not zero, whereas the naive method estimated that its contribution was almost zero, creating a larger gap. In this experiment, because Client 1 and 2 had the same classes of training samples, either one, with smaller data, contributes little to improving the model performance. However, the effect of a model update by the client with little data on validation accuracy at a round was unstable, thereby the client’s update could improve validation accuracy in some rounds, which increases the step-wise contribution metric of the client. This drawback of the proposed method can be mitigated by restricting clients having miniscule amounts of data, which are not expected to improve the global model.
Heuristic 2: 3.55 digits of seven classes. Average errors were Proposed: 1.80, Heuristic 1: 2.55, Heuristic 2: 2.36.

Fig. 5. Errors between Naive metric and each method when Client 2 had digits of five classes. Average errors were Proposed: 1.80, Heuristic 1: 2.55, Heuristic 2: 2.36.

Fig. 6. Errors between Naive metric and each method when Client 2 had digits of seven classes. Average errors were Proposed: 3.57, Heuristic 1: 2.55, Heuristic 2: 3.55.

V. CONCLUSION

This paper proposed an intuitive and computation- and communication-efficient method of estimating the individual contribution levels of participants in FL to determine appropriate incentive mechanisms for FL participation. The proposed method enabled the evaluation during a single FL training process, thereby reducing the need for traffic and computation overhead. The performance evaluations using the MNIST dataset showed that the proposed method estimated the contributions of the individual clients with much smaller computation overhead (plus zero communication overhead) than those of the naive method.

REFERENCES

[1] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, “Machine learning for internet of things data analysis: A survey,” Digital Communications and Networks, vol. 4, no. 3, pp. 161–175, 2018.

[2] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, and J. Qin, “A survey on application of machine learning for internet of things,” International Journal of Machine Learning and Cybernetics, vol. 9, no. 8, pp. 1399–1417, 2018.

[3] P. Voigt and A. Von dem Bussche, “The eu general data protection regulation (gdpr),” A Practical Guide, 1st Ed., Cham: Springer International Publishing, 2017.

[4] R. Shinkuma and T. Nishio, “Data assessment and prioritization in mobile networks for real-time prediction of spatial information with machine learning,” in Proc. IEEE First Workshop on Network Meets Intelligent Computations (NMIC), 2019, pp. 1–6.

[5] Y. Inagaki, R. Shinkuma, T. Sato, and E. Oki, “Prioritization of mobile iot data transmission based on data importance extracted from machine learning model,” IEEE Access, vol. 7, pp. 93 611–93 620, 2019.

[6] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” in Proc. NIPS Workshop on Private Multi-Party Machine Learning, 2016.

[7] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, “A learning-based incentive mechanism for federated learning,” IEEE Internet of Things Journal, 2020.

[8] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, “Robust and communication-efficient federated learning from non-iid data,” IEEE transactions on neural networks and learning systems, 2019.

[9] S. R. Pandey, N. H. Tran, M. Bennis, Y. K. Tun, A. Manzoor, and C. S. Hong, “A crowdsourcing framework for on-device federated learning,” IEEE Transactions on Wireless Communications, vol. 19, no. 5, pp. 3241–3256, 2020.

[10] J. Lee, D. J. Kim, and D. Niyato, “Market analysis of distributed learning resource management for internet of things: A game theoretic approach,” IEEE Internet of Things Journal, 2020.

[11] J. Kang, Z. Xiong, D. Niyato, H. Yu, Y.-C. Liang, and D. I. Kim, “Incentive design for efficient federated learning in mobile networks: A contract theory approach,” in Proc. IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS), 2019, pp. 1–5.

[12] M. Chen, Y. Liu, W. Shen, Y. Shen, P. Tang, and Q. Yang, “Mechanism design for multi-party machine learning,” arXiv preprint arXiv:2001.08996, 2020.

[13] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, “Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory,” IEEE Internet of Things Journal, vol. 6, no. 6, pp. 10 700–10 714, 2019.

[14] Y. Liu, S. Sun, Z. Ai, S. Zhang, Z. Liu, and H. Yu, “Fedcoin: A peer-to-peer payment system for federated learning,” arXiv preprint arXiv:2002.11711, 2020.

[15] K. Toyoda and A. N. Zhang, “Mechanism design for an incentive-aware blockchain-enabled federated learning platform,” in Proc. IEEE International Conference on Big Data (Big Data), 2019, pp. 395–403.

[16] H. Yu, Z. Liu, Y. Liu, T. Chen, M. Cong, X. Weng, D. Niyato, and Q. Yang, “A fairness-aware incentive scheme for federated learning,” in Proc. AAAI/ACM Conference on AI, Ethics, and Society, 2020, pp. 393–399.

[17] ——, “A sustainable incentive scheme for federated learning,” IEEE Intelligent Systems, 2020.

[18] W. Y. B. Lim, Z. Xiong, C. Miao, D. Niyato, Q. Yang, C. Leung, and H. V. Poor, “Hierarchical incentive mechanism design for federated machine learning in mobile networks,” IEEE Internet of Things Journal, 2020.

[19] Y. Jiao, P. Wang, D. Niyato, B. Lin, and D. I. Kim, “Toward an automated auction framework for wireless federated learning services market,” IEEE Transactions on Mobile Computing, 2020.

[20] R. Zeng, S. Zhang, J. Wang, and X. Chu, “Fmoore: An incentive scheme of multi-dimensional auction for federated learning in mec,” arXiv preprint arXiv:2002.09699, 2020.

[21] T. Nishio and R. Yonetani, “Client selection for federated learning with heterogeneous resources in mobile edge,” in Proc. IEEE International Conference on Communications (ICC), 2019, pp. 1–7.

[22] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[23] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Irving, M. Isard, M. Kudlur, L. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: A system for large-scale machine learning,” in Proc. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, GA, Nov. 2016, pp. 265–283.