rFedKD: A Reverse Federated Knowledge Distillation Method for Communication Efficiency

Weijia Feng[1], Ruojia Zhang[1], Yichen Zhu[1], Chenyang Wang[2](✉), Xiaobao Wang[3], Tarik Taleb[4]

[1] Tianjin Normal University, Tianjin 300387, China
weijiafeng@tjnu.edu.cn,{zrj20001127, zhuyiiichen}@163.com
[2] Shenzhen University, Shenzhen 518060, China
chenyangwang@ieee.org
[3] Tianjin University, Tianjin 30035
wangxiaobao@tju.edu.cn
[4] Ruhr University Bochum, 44801 Bochum, Germany
tarik.taleb@rub.de

Abstract. Artificial intelligence (AI) rapidly advances technological innovation, particularly in data processing and intelligent decision-making. Edge computing (EC) addresses key challenges in AI deployment, such as reducing latency and reliance on centralized cloud infrastructure by enabling processing at the network edge. However, EC faces limitations in managing computational complexity, latency, and resource constraints on edge devices. To overcome these challenges, we propose a novel reverse Federated Knowledge Distillation (rFedKD) method. Unlike traditional knowledge distillation, rFedKD extracts knowledge from small, personalized models at the edge and integrates it into a large central model. This approach aggregates diverse information while maintaining client-specific knowledge. The central model, leveraging its generalization capabilities, improves the accuracy of personalized edge models and accelerates training processes. The experimental results demonstrate the effectiveness of rFedKD, achieving a 15% improvement in accuracy and reducing communication rounds by 20 compared to the latest methods. This enhances system efficiency and user experience, establishing rFedKD as a promising solution for advancing edge AI.

Keywords: Artificial Intelligence · Edge Computing · Knowledge Distillation · Federated Learning.

1 Introduction

The demand for high-quality content has driven the adoption of AI-generated content (AIGC) to address digital economy challenges, with applications in natural language generation and image creation (e.g., ChatGPT [1], Stable Diffusion [2]). AIGC uses AI algorithms for tasks like text and image generation, while edge computing (EC) deploys models on edge servers to reduce latency [4–6]. However, EC faces computational complexity, latency, and resource constraints on edge devices [7–9]. Traditional knowledge distillation (KD) transfers knowledge from large to small models, but federated knowledge distillation (FedKD) often requires significant resources and bandwidth [10].

To address these challenges, we propose a Reverse Federated Knowledge Distillation (rFedKD) framework, which distills knowledge from small edge models into a large central model. This simplifies training into two stages: UE-side training and server-side aggregation, reducing resource consumption and complexity while improving accuracy and speed. The main contributions are: 1) Reverse Knowledge Distillation, integrating knowledge from small models into a large model to enhance performance. 2) Elimination of Model Aggregation, allowing the server to transmit smaller, efficient models. 3) Improved System Performance, with experiments showing 15% higher accuracy and 20% fewer communication rounds compared to state-of-the-art methods.

2 Related Work

AIGC techniques have been widely applied to network resource coordination. For instance, Tang *et al.* [11] proposed an automated, distributed, AI-enabled testing framework that employs a master-actor architecture to manage terminal devices for distributed testing. The framework utilizes AI to intelligently explore the decision space of AI models in O-RAN, facilitating the evaluation of decision performance, vulnerabilities, and security. Alhammadi *et al.* [12] surveyed AI techniques across various wireless networks and applications, highlighting unresolved research challenges and proposing potential solutions. Ioannou *et al.* [13] introduced a distributed AI framework based on machine learning, incorporating modular BDIx agents that extend the belief-desire-intent (BDI) architecture with ML capabilities. However, these works mainly address the deployment and efficiency of AI models without focusing on the specific challenges of KD in federated settings, leading to the over-consumption of the system resources.

Multiple studies have explored FedKD methods to address edge intelligence challenges. Matsubara *et al.* [14] modified DNN structures for image classification, achieving high accuracy via KD-based compression. Ma *et al.* [15] proposed CFeD to address FL's forgetting problem using KD on UEs and servers with unlabeled data. Chadha *et al.* [16] extended FedLess to reduce resource and statistical heterogeneity in FL. Li *et al.* [17] introduced ECCT for bidirectional knowledge transfer between edge and cloud. Wang *et al.* [18] developed Shoggoth, an edge-cloud collaboration architecture that improves accuracy by 15%-20% and reduces network costs. Zhao *et al.* [19] proposed MESON for urban VEC, using DRL to enhance response time and energy efficiency. Xu *et al.* [20] analyzed security and privacy challenges in mobile AIGC networks. However, these methods often involve complex model aggregation and multiple training stages, increasing computational and communication overhead.

3 System Model

We consider that the system comprises multiple user equipment (UEs) $\mathcal{K} = \{k|1, 2, \cdots, K\}$, and base stations (BSs) $\mathcal{B} = \{b|1, 2, \cdots, B\}$ with limited storage resources $\mathcal{C}_b \in R^+$. The cloud C maintains a model library with $\mathcal{M} = \{m|1, 2, \cdots, M\}$ popular models, each of size $\kappa_m \in \mathbb{R}^+$. Users maintain relatively fixed positions over a period $\mathcal{T} = \{t|1, 2, \cdots, T\}$, enabling local and server model training. Each BS is connected to neighboring BSs via wired links and to the cloud via backhaul links. UEs collect and cache data, generating computation tasks $\mathcal{P} = \{P_1, P_2, \cdots, P_K\}$, and maintains private datasets $\mathcal{D}_k, k \in \mathcal{K}$. TThe edge server (ES) determines offloading strategies for tasks $\mathcal{P}$ based on collected information $\mathcal{I}_k$ and environmental conditions. The computation load of P_k depends on the device's CPU cycles ρ_k. Due to the small size of model $\mathcal{M}$ and the proximity of UEs to ESs, latency and energy costs for model exchanges are negligible.

3.1 Task Transmission and Computation Model

Tasks P can be executed locally or offloaded to the edge server (ES) or cloud, denoted as $\mathbf{a}_P = \{a_P^k, a_P^b, a_P^C\}$, where $a_P^k, a_P^b, a_P^C \in \{0, 1\}$ indicate execution at the local UE, ES, or cloud, respectively, with the constraint:

$$\mathcal{C}1 : a_P^k + a_P^b + a_P^C = 1, \forall P \in \mathcal{P}, \tag{1}$$

Each UE generates a dataset $\mathcal{D}_k$ and trains the model locally. The wireless link rate between UE k and ES b is

$$v_{k,b}^t = \mathbf{w}_{k,b}^t log_2(1 + \frac{q_b|h_{k,b}|^2}{\sigma^2}), \forall k \in \mathcal{K}, b \in \mathcal{B}, t \in \mathcal{T}, \tag{2}$$

where $\mathbf{w}_{k,b}^t = (w_{k,b}^t)_{k \in \mathcal{K}}$ are the wireless bandwidth allocated to the users, W_b is the total channel bandwidth, q_b is the assigned transmit power of b, $h_{k,b}$ denotes the channel gain, and σ^2 is the noise power. The transmission latency for uploading task P to the ES is $l_{P,KB}^{trans} = \frac{|\mathcal{D}_k|}{v_{k,b}^t}$. and the transmission latency from ES to the cloud is $l_{P,BC}^{trans} = \frac{|\mathcal{D}_k|}{v^C}$, where v_C is the average transmission rate of each task from the ES to the cloud.

When task P is executed locally, *i.e.*, on the UE k, recall that the average CPU frequency of UE k is ρ_k, then the computation latency is expressed as $l_{P,k}^{comp} = \frac{|\mathcal{D}_k|}{\rho_k}$. similarly, the edge execution delay can be obtained as $l_{P,b}^{comp} = \frac{|\mathcal{D}_k|}{\rho_b}$, where ρ_b is the computing speed of ES b. Finally, the edge execution delay can be obtained as $l_{P,C}^{comp} = \frac{|\mathcal{D}_k|}{\rho_C}$, where the ρ_C is the average computing speed for each task at the cloud C.

3.2 Knowledge Distillation Model

Each UE has its own dataset $\mathcal{D}_k$, which is transmitted to the server and aggregated into a distillation dataset $\mathcal{D} = \{\mathcal{D}_k|k \in \mathcal{K}\}$. After E local training rounds, each UE model outputs predictions S_t^k and test accuracy Acc_k on $\mathcal{D}$. The server model also outputs predictions S_t^s on the same dataset. Each UE then integrates its own predictions with the server's predictions, evaluates their correlation and accuracy, and generates a weight $\hat{\alpha}_k$. Using these weights, the UE computes its overall prediction S_t^c by summing the weighted predictions:

$$S_t^c = \sum_{k=1}^{\mathcal{K}} \hat{\alpha}_k S_t^k, \forall t \in \mathcal{T}. \tag{3}$$

Knowledge Distillation (KD) trains a student model using soft logits from the teacher and student models on dataset $\mathcal{D}$, involving task loss and knowledge loss. involving task loss and knowledge loss. In our framework, multiple UE models act as the teacher, and the server model as the student, with outputs S_t^s and S_t^c. The KD loss with distillation temperature θ_{kd} is:

$$L_{kd}(W_t^s) = \beta * L_{CE}(S_t^s, y_{\mathcal{D}}) + (1-\beta) L_{KL}(S_t^s, S_t^c) * \theta_{kd}^2, \forall t \in \mathcal{T}, \tag{4}$$

where W_t^s is the student model weight, $y_{\mathcal{D}}$ is the ground truth, and β is a balance weight of classification loss and distillation loss.

4 Propsoed rFedKD Framework

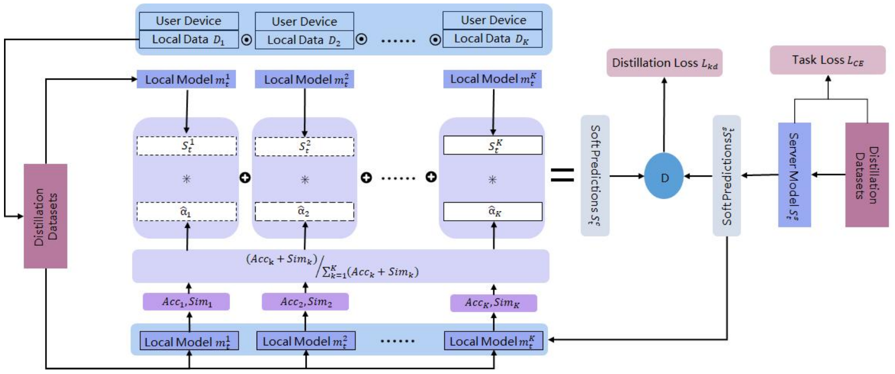


Fig. 1. Training Process of rFedKD

Overview of rFedKD The whole process of the $t-th$ training round in the proposed framework is shown in Fig.1. In our proposed rFedKD framework, the training process in the $t-th$ round involves three main stages. First, the server selects $\mathcal{K}$ active UEs based on an activation ratio δ, which uses their private data D_k to train local models. These models' predictions S_t^k are sent to the server. Second, the server calculates weights $\hat{\alpha}_k$ for each UE based on the cosine similarity Sim_k and accuracy Acc_k between the UE and server predictions. Finally, the server updates its model using a weighted sum of UE predictions and computes the distillation loss $L_{kd}(W_t^s)$. Finally, the server model can be updated based on $L_{kd}(W_t^s)$.

Local User Model Training In the local user model training phase, we employ the ResNet model as the backbone for UEs and set the total local training steps to E. Each UE updates its local model by minimizing the cross-entropy loss function $L(f(x_k, m_e^k), y_k) = -\sum_{i=1}^{I} y_k^i log(f(x_k^i, m_e^k))$, where $f(x_k^i, m_e^k)$ represents the convolution operation within the residual block, and W_e^k is the weight matrix of the local model at training step e. The local model is initialized using the model from the last communication round, $m_0^k = m_{t-1}^k$. At each step e, the UE updates the model via backpropagation:

$$\begin{aligned} & g_t = \nabla_t L(f(x_{D_k}, m_e^k), y_{D_k}), \\ & m_{e+1}^k = m_e^k - \eta g_e = m_e^k - \nabla_e L(f(x_{D_k}, m_e^k), y_{D_k}), \end{aligned} \tag{5}$$

and the final local model is $m_t^k = m_E^k$.

Weighted Factors Calculating After local training, each UE generates predictions S_t^k on its dataset $\mathcal{D}_k$. The server computes its own predictions S_t^s on the same dataset. Based on the cosine similarity $Sim_k = F_c(f(x_{\mathcal{D}}, m_t^k), f(x_{\mathcal{D}}, m_t^s))$ and accuracy $Acc_k = F_k(f(x_{\mathcal{D}}, m_t^k), y_{\mathcal{D}})$ between the server's and UE's predictions, the server calculates a weight $\hat{a}_k$ for each UE as

$$\hat{a}_k = \frac{Sim_k + Acc_k}{\sum_{k=1}^{K}(Sim_k + Acc_k)}, k \in \{1, 2, \cdots, K\}, \tag{6}$$

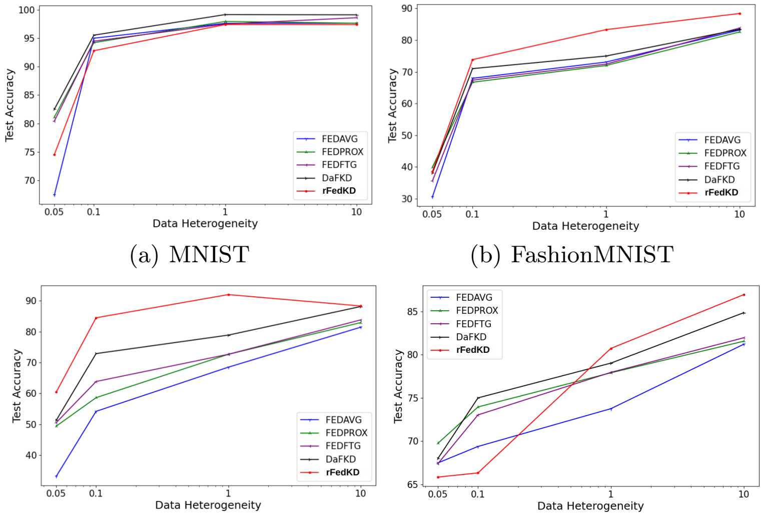
These weights reflect the correlation and accuracy of each UE's model, allowing the server to aggregate the predictions effectively.

Server Model Distillation Through the first two steps, we obtain each UE's predictions and weights, along with the server's predictions. Using traditional KD, the server model is updated by treating the federated UEs as the teacher and the server as the student. The teacher's knowledge is derived by calculating the weighted sum of predictions from the UEs, which is then used to update the server model. The process is outlined as follows:

$$\begin{aligned} & W_{t+1}^s = W_t^s - \nabla_t L_{kd}(W_t^s) \\ & = W_t^s - \nabla_t \{\delta * L_{CE}(S_t^s, y_{\mathcal{D}}) + (1-\delta) L_{KL}(S_t^s, S_t^c) * {\theta_{kd}}^2\}. \end{aligned} \tag{7}$$

5 Experiments

For experimental setups, we compared rFedKD with FEDAVG, FEDPROX, FEDDFUSION, FEDFTG, and DaFKD. These methods were chosen for their diverse approaches to federated learning and knowledge distillation, providing comprehensive benchmarks. Experiments used MNIST, FashionMNIST, SVHN,



(a) MNIST

(b) FashionMNIST

(c) SVHN

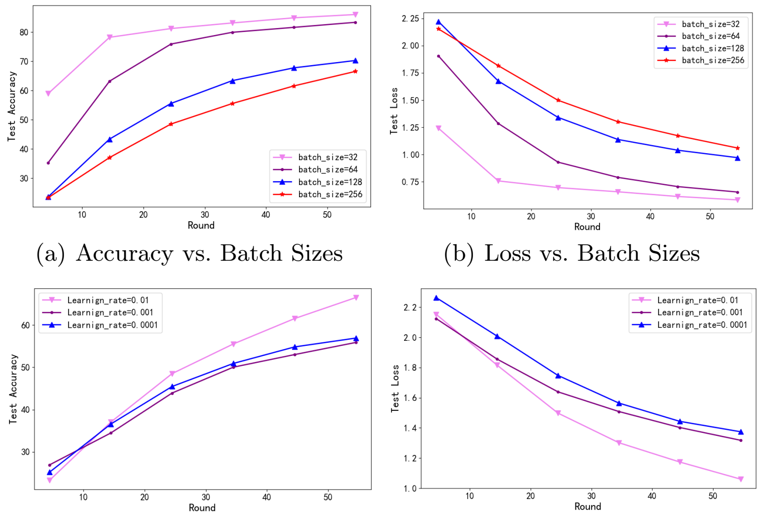
(d) EMNIST

Fig. 2. Accuracy performance with different data heterogeneity on four image datasets

and EMNIST to evaluate performance across different data types. The simulation configuration included 20 local training epochs (E=20), 60 communication rounds (T=60), and 20 UEs with an activation ratio of 0.2. Local training used a batch size of 32, a weight decay of 0.001, and a learning rate of 0.01 for distillation. Data heterogeneity was simulated using a Dirichlet distribution Dir(α), with all training data assigned to user models and test samples used for evaluation. ResNet8 was used as the UE-side model and ResNet11 as the server-side model to reflect the limitations of the device and the server capabilities.

Table 1. Top-1 Test Accuracy

Datasets	Settings	FEDFVG	FEDPROX	FEDFTG	DaFKD	rFedKD
MNIST, E=20	α=0.05	69.11 ± 1.39	80.77 ± 0.35	80.95 ± 1.06	**82.33 ± 0.44**	75.90 ± 1.27
	α=0.1	95.16 ± 0.79	93.21 ± 0.55	94.43 ± 0.49	**95.56 ± 0.41**	91.88 ± 2.79
	α=1	98.11 ± 0.14	97.08 ± 0.69	98.47 ± 0.21	**98.96 ± 0.38**	98.08 ± 0.19
FashionMNIST, E=20	α=0.05	30.01 ± 0.54	**39.71 ± 0.23**	34.84 ± 0.77	37.85 ± 0.24	38.59 ± 1.00
	α=0.1	67.97 ± 0.03	66.65 ± 0.08	67.25 ± 0.14	70.81 ± 0.21	**71.89 ± 2.30**
	α=1	82.37 ± 0.82	82.06 ± 0.53	81.96 ± 1.86	83.49 ± 1.32	**87.92 ± 0.43**
SVHN, E=20	α=0.05	33.01 ± 0.12	49.24 ± 0.16	48.69 ± 1.87	51.14 ± 0.16	**60.23 ± 0.20**
	α=0.1	53.54 ± 0.21	57.77 ± 0.86	63.75 ± 0.11	72.80 ± 0.11	**84.83 ± 0.21**
	α=10	81.44 ± 0.01	82.61 ± 0.34	83.49 ± 1.32	87.31 ± 0.85	**88.30 ± 0.06**
EMNIST, E=40	α=0.05	67.28 ± 0.14	69.73 ± 0.17	67.08 ± 0.97	67.64 ± 1.86	**68.71 ± 1.58**
	α=0.1	69.13 ± 0.23	73.72 ± 0.55	72.91 ± 1.87	**74.96 ± 0.91**	69.83 ± 3.46
	α=10	81.35 ± 1.03	81.61 ± 0.71	82.65 ± 1.04	84.60 ± 1.04	**86.40 ± 0.54**



(a) Accuracy vs. Batch Sizes

(b) Loss vs. Batch Sizes

(c) Accuracy vs. LR

(d) Loss vs. LR

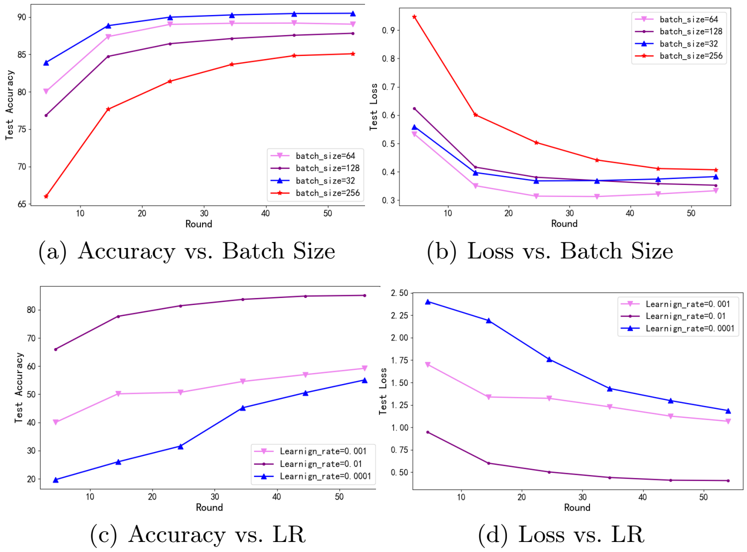
Fig. 3. Accuracy and loss of different batch sizes and learning rates on the SVHN dataset

Top-1 Tests Accuracy We evaluated the Top-1 test accuracy of rFedKD compared to other baseline methods on five image classification datasets, and the results are shown in Table 1. rFedKD performs well on all datasets, especially in cases of high data heterogeneity (with small α values). For example, on the SVHN dataset, the accuracy of rFedKD is 2.09% to 15% higher than the second-best method at different levels of heterogeneity. These results indicate that rFedKD can effectively utilize the correlation between user devices and servers to distill knowledge from multiple smaller models into a larger server model, thereby improving accuracy.

Communication Rounds We use the number of communication rounds required to achieve a target accuracy as the QoE evaluation criterion. For each dataset, we set one or two target accuracy rates and compare the rounds needed by each model. Table 2 shows that rFedKD achieves the best results across all datasets, requiring the fewest rounds to reach the target accuracy while also achieving the highest test accuracy under varying heterogeneity levels (α).

Data Heterogeneity We also analyzed the performance of various methods under different levels of data heterogeneity, as shown in Figure 2. As data heterogeneity increases (α value decreases), the performance advantage of rFedKD becomes more apparent. For example, on the CIFAR-10 dataset, rFedKD showed significantly higher accuracy than other methods in cases of high data heterogeneity (α=0.05). This indicates that rFedKD is robust when dealing with scenarios with significant differences in the distribution of user data.

We conducted experiments on the SVHN and FashionMNIST datasets to evaluate the sensitivity of rFedKD to specific parameters by varying the sam-



(a) Accuracy vs. Batch Size

(b) Loss vs. Batch Size

(c) Accuracy vs. LR

(d) Loss vs. LR

Fig. 4. Accuracy and loss of different batch sizes and learning rates on the FashionMNIST dataset

Table 2. Communication Rounds That Achieve the Specified Accuracy

Dataset	Accuracy	FEDAVG	FEDPROX	FEDFTG	DaFKD	rFedKD
MNIST,	*acc*=85%	25	22	24	23	**12**
E=20	*acc*=90%	34s	43	43	**40**	45
FashionMNIST,	*acc*=60%	23	29	33	21	**4**
E=20	*acc*=65%	40	43	50	37	**6**
SVHN,	*acc*=55%	>60	54	35	17	**5**
E=20	*acc*=60%	>60	>60	58	20	**10**
EMNIST,	*acc*=65%	23	22	27	24	**19**
E=40	*acc*=70%	59	48	42	45	**30**

ple batch size (B) and distillation learning rate (l_d). As shown in Figure 5 and Figure 5, the experimental results indicate that smaller batch sizes (such as 32) can achieve higher accuracy and lower testing losses on the SVHNand FashionMNIST datasets, and converge faster. In addition, we also tested different distillation learning rates (0.01, 0.001, and 0.0001), and the results showed that higher learning rates can achieve faster convergence and higher accuracy within a limited training period. These experimental results indicate that rFedKD is sensitive to parameter settings, and appropriate parameter selection can significantly improve model performance.

6 Conclusion

This study has proposed rFedKD, a federated knowledge distillation framework that eliminates traditional model aggregation and supports diverse model architectures across devices. Using a dynamic, accuracy-based weighting mechanism, rFedKD has enhanced performance in heterogeneous environments, ensuring higher-accuracy models contribute more significantly to the global model. Experimental results have validated rFedKD's competitive accuracy, robustness, and scalability across datasets. The framework's dynamic weight adjustment has promoted efficient model collaboration, offering a promising solution for federated learning in diverse scenarios. Future work will further optimise weight parameters and apply the framework to other domains.

7 Acknowledgments

This study is supported in part by NSFC (Natural Science Foundation of China): 61602345,62002263, in part by National Key Research and Development Plan: 2019YFB2101900, in part by TianKai Higher Education Innovation Park Enterprise R&D Special Project (23YFZXYC00046).

References

1. Wang, Y., Pan, Y., Yan, M., Su, Z., and Luan, T. H.: A survey on ChatGPT: AI-generated contents, challenges, and solutions. IEEE Open Journal of the Computer Society (2023)
2. Yi, Y., Zhang, Z., Yang, L. T., Deng, X., Yi, L., and Wang, X.: Social interaction and information diffusion in social Internet of Things: Dynamics, cloud-edge, traceability. IEEE Internet of things Journal **8**(4), 2177-2192 (2020)
3. Wang, X., Wang, C., Li, X., Leung, V. C., and Taleb, T.: Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching. IEEE Internet of Things Journal **7**(10), 9441-9455 (2020)
4. Sun, C., Li, X., Wang, C., He, Q., Wang, X., and Leung, V. C.: Hierarchical Deep Reinforcement Learning for Joint Service Caching and Computation Offloading in Mobile Edge-Cloud Computing. IEEE Transactions on Services Computing (2024)
5. Wang, C., Yu, H., Li, X., et al. Dependency-Aware Microservice Deployment for Edge Computing: A Deep Reinforcement Learning Approach with Network Representation[J]. IEEE Transactions on Mobile Computing, 2024.
6. Zhu, Z., Hong, J., and Zhou, J.: Data-free knowledge distillation for heterogeneous federated learning. In: International conference on machine learning, pp. 12878-12889 (2021)
7. Wu, C., Wu, F., Lyu, L., Huang, Y., and Xie, X.: Communication-efficient federated learning via knowledge distillation. Nature communications **13**(1), pp. 2023 (2022)
8. Wang, X., Wang, C., Li, X., Leung, VCM., Taleb, T.: Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching. In: IEEE Internet of Things Journal **7**(10), pp. 9441–9455 (2020)
9. Lee, G., Jeong, M., Shin, Y., Bae, S., and Yun, S. Y.: Preservation of the global knowledge by not-true distillation in federated learning. Advances in Neural Information Processing Systems 35, 38461-38474 (2022)

10. Yang, X., Yu, H., Gao, X., and *et al.*: Federated continual learning via knowledge fusion: A survey. IEEE Transactions on Knowledge and Data Engineering (2024)
11. Tang, B., Shah, V. K., Marojevic, V., and Reed, J. H.: AI testing framework for next-G O-RAN networks: Requirements, design, and research opportunities. IEEE Wireless Communications **30**(1), 70-77 (2023)
12. Alhammadi, A., Shayea, I., El-Saleh, A. A., Azmi, M. H., Ismail, Z. H., Kouhalvandi, L., and Saad, S. A.: Artificial Intelligence in 6G Wireless Networks: Opportunities, Applications, and Challenges. International Journal of Intelligent Systems 2024 (1), 8845070 (2024)
13. Ioannou, I., Christophorou, C., Vassiliou, V., and Pitsillides, A.: A novel Distributed AI framework with ML for D2D communication in 5G/6G networks. Computer Networks 211, 108987 (2022)
14. Matsubara, Y., Callegaro, D., Baidya, S., Levorato, M., and Singh, S.: Head network distillation: Splitting distilled deep neural networks for resource-constrained edge computing systems. IEEE Access **8**, 212177-212193 (2020)
15. Ma, Y., Xie, Z., Wang, J., Chen, K., and Shou, L.: Continual Federated Learning Based on Knowledge Distillation. In: International Joint Conference on Advances in Computational Intelligence (IJCAI), pp. 2182-2188 (2022)
16. Chadha, M., Khera, P., Gu, J., Abboud, O., and Gerndt, M.: Training Heterogeneous Client Models using Knowledge Distillation in Serverless Federated Learning. In: Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing, pp. 997-1006 (2024)
17. Li, Z., Li, Q., Zhou, Y., Zhong, W., Zhang, G., and Wu, C.: Edge-cloud collaborative learning with federated and centralized features. In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1949-1953 (2023)
18. Wang, L., Lu, K., Zhang, N., and *et al.*: Shoggoth: towards efficient edge-cloud collaborative real-time video inference via adaptive online learning. In: 2023 60th ACM/IEEE Design Automation Conference (DAC), pp. 1-6 (2023)
19. Zhao, L., Zhang, E., Wan, S., and *et al.*: MESON: A mobility-aware dependent task offloading scheme for urban vehicular edge computing. IEEE Transactions on Mobile Computing (2023)
20. Xu, M., Du, H., Niyato, D., Kang, J., Xiong, Z., Mao, S., and Poor, H. V.: Unleashing the power of edge-cloud generative ai in mobile networks: A survey of aigc services. IEEE Communications Surveys & Tutorials (2024)