Edge-cloud Collaborative Learning with Federated and Centralized Features

Zexi Li* Zhejiang University Hangzhou, China zexi.li@zju.edu.cn

Wenliang Zhong* Ant Group Hangzhou, China yice.zwl@antgroup.com Qunwei Li* Ant Group Hangzhou, China qunwei.qw@antgroup.com

Guannan Zhang Ant Group Hangzhou, China zgn138592@antgroup.com Zhou Yi The University of Utah Salt Lake City, USA yi.zhou@utah.edu

Chao Wu^{*} Zhejiang University Hangzhou, China chao.wu@zju.edu.cn

ABSTRACT

Federated learning (FL) is a popular way of edge computing that doesn't compromise users' privacy. Current FL paradigms assume that data only resides on the edge, while cloud servers only perform model averaging. However, in real-life situations such as recommender systems, the cloud server has the ability to store historical and interactive features. In this paper, our proposed Edge-Cloud Collaborative Knowledge Transfer Framework (ECCT) bridges the gap between the edge and cloud, enabling bi-directional knowledge transfer between both, sharing feature embeddings and prediction logits. ECCT consolidates various benefits, including enhancing personalization, enabling model heterogeneity, tolerating training asynchronization, and relieving communication burdens. Extensive experiments on public and industrial datasets demonstrate ECCT's effectiveness and potential for use in academia and industry.

CCS CONCEPTS

• Computing methodologies → Machine learning; • Information systems → Data mining.

KEYWORDS

Edge-cloud collaborative learning, Federated learning, Recommender systems

ACM Reference Format:

Zexi Li, Qunwei Li, Yi Zhou, Wenliang Zhong, Guannan Zhang, Chao Wu. 2023. 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 (SIGIR '23), July 23–27, 2023, Taipei, Taiwan. ACM, New York, NY, USA, 5 pages. https://doi.org/10. 1145/3539618.3591976

* Corresponding authors

SIGIR '23, July 23–27, 2023, Taipei, Taiwan.

1 INTRODUCTION

In the conventional centralized training paradigm, data is collected from multi-sourced edge devices, based on which a large model is trained at a cloud server [16, 21]. Recently, due to user privacy and communication efficiency concerns, federated learning (FL) has been proposed as an alternative to the centralized training paradigm [13–15, 18, 24]. FL assumes all the training data is kept locally at the edge devices, and a global model can be obtained by iterative edge training and cloud-side model averaging. It has shown feasibility and applicability in privacy-preserving scenarios. For example, keyboard inputs are sensitive information of cellphone users, and Google has used FL to improve next-word prediction while preserving users' privacy [4].

However, the current FL frameworks ignore the existence of cloud-side data, and it does not directly incorporate cloud training. In industrial applications, the cloud server usually has rich computation resources to train a big model with rich historical and non-private features. Taking the recommender system (RS) as an example, privacy-sensitive features, including one user's gender, age, and other real-time features such as the user's timely interactions with the items are generally inaccessible to the cloud and are stored locally in the devices. We denote these features as "federated features". On the other hand, the centralized historical features (with fewer privacy concerns), including the user's historical interactions with items, item categories, item flags, item-to-item embeddings, etc., are stored on the cloud server. We name these features as "centralized features". We find that the existing FL frameworks do not fully utilize these two sets of features. The main reason is that the federated and centralized features have different feature spaces, dimensions, and accessibility, and it is therefore challenging to utilize the federated features to improve the global model's performance. Also, it is impractical to move the centralized features to the edge since such features are large in size and the edge devices have limited storage and computation resources.

The conventional centralized learning paradigm and the current FL paradigm only utilize one-sided cloud training or edge training. However, we reckon the edge-cloud collaboration is quite important in practice to have both better-generalized representations and more personalized user services [9, 20, 27]. Thus, in this paper, we propose a Edge-Cloud Collaborative Knowledge Transfer Framework (ECCT). In this framework, we train a big model on the server and a light weighted model on each edge device. The

^{*} Both authors contributed equally to this work.

Work was done when Zexi Li was an intern at Ant Group.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

^{© 2023} Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 978-1-4503-9408-6/23/07...\$15.00 https://doi.org/10.1145/3539618.3591976

feature embeddings and prediction logits are shared between the edge and cloud to support knowledge transfer and collaboration. The edge and cloud models then adopt an alternating minimization (AM) approach [2, 3, 10] to utilize the transferred knowledge via embedding fusion and knowledge distillation.

ECCT offers several advantages over conventional FL frameworks. Firstly, it allows for boosting local personalization by transferring knowledge from centralized features to on-device models. This improves personalization compared to FL methods. Secondly, ECCT enables model heterogeneity, allowing edge models to differ in architecture while still being compatible. Thirdly, ECCT naturally supports asynchronous training and is more robust when a partial selection of training devices exists. Finally, ECCT relieves communication burdens because transferred knowledge has a much smaller size than model parameters.

Our contributions are as follows.

- We develop the ECCT framework that jointly utilizes the cloud's centralized features and the edge's federated features. To the best of our knowledge, this is the first paper to use the two kinds of features to realize edge-cloud collaboration, and it has promising industrial applications.
- In ECCT, we implement bi-directional knowledge transfers between the edge and cloud by embedding fusion and knowledge distillation.
- Extensive experiments under both public and industrial datasets demonstrate the effectiveness of ECCT and show its broad and practical prospects in the industry.

2 RELATED WORKS

Edge-cloud collaborative federated learning. FedGKT [10] incorporates split learning in FL to realize edge-cloud collaboration. It trains a larger CNN model on the server based on the embeddings and logits from the devices. However, it does not utilize centralized data, and the knowledge from the cloud to the edge is weak by just transferring logits. There are works that take the cloud-side data as a public dataset to aid edge training via knowledge distillation [7, 19, 28]. We reckon it is not realistic to store such a public dataset at the edge devices, which hinders their applications in the industry. Edge-cloud collaborative recommender systems. In [27], Mo-MoDistill is proposed to finetune the meta patches of the cloud RS model at the edge, and it can realize user personalization. In [26], a causal meta controller is trained to manage the tradeoff between the on-device and the cloud-based recommendations. We note that [26, 27] assume the centralized data is the whole sum of the federated data. But in realistic scenarios, the centralized and federated data have different features due to privacy and real-time issues. There are methods solving the inference problems in edge recommendation in terms of re-ranking [9] and item request [20].

3 METHODOLOGY

In Section 3.1, we give the main formulation of the proposed framework, while in Section 3.2, we provide several practical add-ons that can be applied with the framework to further strengthen performance or privacy guarantees.

Li and Li, et al.

3.1 Main Formulation

Problem Setup. There are *K* edge devices in the network. Specifically, the learning task is supervised learning for classification with *C* categories in the entire dataset, which consists of two parts of features, geographically located at the server (centralized features, \mathcal{D}) and the devices (federated features, $\hat{\mathcal{D}}$). The *k*-th device has its

own dataset of private and real-time features $\hat{\mathcal{D}}^k := \left\{ \left(\hat{\mathbf{X}}_i^k, y_i^k \right) \right\}_{i=1}^{N^k}$, where $\hat{\mathbf{X}}_i$ is the *i*-th training sample, y_i is the corresponding label of $\hat{\mathbf{X}}_i, y_i^k \in \{1, 2, ..., C\}$, and N^k is the sample size of dataset $\hat{\mathcal{D}}^k$ (also D^k). Thus, we have $\hat{\mathcal{D}} = \{\hat{\mathcal{D}}^1, \hat{\mathcal{D}}^2, ..., \hat{\mathcal{D}}^K\}$, $N = \sum_{k=1}^K N^k$. The server has all non-private and historical features of all devices

as
$$\mathcal{D}^k \coloneqq \left\{ \left(\mathbf{X}_i^k, y_i^k \right) \right\}_{i=1}^{N^*}, \mathcal{D} = \{ \mathcal{D}^1, \mathcal{D}^2, \dots, \mathcal{D}^K \}$$

Learning Objectives. For edge device k, we deploy a small feature encoder (extractor) \mathcal{E}_d^k and a small classifier C_d^k , while for the cloud server, we deploy a large encoder \mathcal{E}_s and a large-scale downstream model C_s (including the classifier). Let $\mathcal{W}_d^k = \{\mathcal{E}_d^k, C_d^k\}$ and $\mathcal{W}_s = \{\mathcal{E}_s, C_s\}$. Due to the separated features at the cloud and edge, we reformulate a single global model optimization into a non-convex optimization problem that requires us to train the server model \mathcal{W}_s and the device model \mathcal{W}_d^k simultaneously. The learning objectives of the server and device k with respective loss functions of ℓ_s and ℓ_d^k are as follows

$$\arg\min_{\mathcal{W}_s} F_s(\mathcal{W}_s) = \arg\min_{\mathcal{W}_s} \sum_{k=1}^K \sum_{i=1}^{N^k} \ell_s \left(C_s(h_{d,i}^k \oplus h_{s,i}^k), y_i^k \right), \quad (1)$$

$$\arg\min_{\mathcal{W}_d^k} F_d^k(\mathcal{W}_d^k) = \arg\min_{\mathcal{W}_d^k} \sum_{i=1}^N \ell_d^k \left(C_d^k(h_{d,i}^k \oplus h_{s,i}^k), y_i^k \right),$$
(2)

NTK.

where
$$h_{d,i}^k = \mathcal{E}_d^k(\hat{\mathbf{X}}_i^k), \ h_{s,i}^k = \mathcal{E}_s(\mathbf{X}_i^k).$$
 (3)

In Eqs. 1 and 2, $h_{d,i}^k$ $(h_{s,i}^k)$ refers to the extracted feature embedding of a sample $\hat{\mathbf{X}}_i^k$ (\mathbf{X}_i^k) using the device's (server's) encoder \mathcal{E}_d^k (\mathcal{E}_s) and \oplus denotes the concatenation operation that concatenates two embeddings into one. We optimize the above objectives via alternating minimization (AM) approach [2, 3, 10]. For the server's optimization, we fix the devices' embeddings $h_{d,i}^k$, and for device k's optimization, we fix the server's embedding $h_{s,i}^k$.

Apart from the embeddings, we also transfer the prediction logits and adopt knowledge distillation (KD) [11, 17] to strengthen knowledge learning. The shared embeddings and logits realize a bi-directional knowledge transfer in the edge-cloud collaborative training, and we adopt the loss functions for the server and edge device k respectively as follows

$$\ell_s = \ell_{CE} + \alpha_s \sum_{k=1}^K \ell_{KD} \left(z_d^k, z_s^k \right), \tag{4}$$

$$\ell_d^k = \ell_{CE}^k + \alpha_d \ell_{KD}^k \left(z_s^k, z_d^k \right), \tag{5}$$

where ℓ_{CE} is the cross-entropy loss between the predicted values and the ground truth labels, ℓ_{KD} is the Kullback Leibler Divergence function for knowledge distillation, and z_s^k and z_d^k are prediction logits. Besides, α_s and α_d are the hyper-parameters to control the strengths of knowledge distillation. Edge-cloud Collaborative Learning with Federated and Centralized Features



Figure 1: The proposed edge-cloud collaborative knowledge transfer framework.

Training Workflow. The demonstration of the proposed framework is shown in Figure 1. There are mainly three iterative steps in our edge-cloud collaborative learning framework: edge training, cloud training, and edge-cloud collaboration. (1) *Cloud training*. The cloud server conducts cloud training as that of Eq. 4 for E_s epochs. (2) *Edge training*. The edge device *k* conducts edge training as that of Eq. 5 for E_d^k epochs. (3) *Edge-cloud communication*. The edge device (cloud server) infers the embeddings via the encoder and the logits via the classifier on its local features and sends the embeddings and logits to the cloud server (edge devices).

Steps 1 and 2 can be conducted at the cloud and edge in parallel, and the AM-based method makes our framework more tolerant to asynchronous updates and communications. In practice, the device model accumulates its embeddings in a buffer, and when the buffer is full, it communicates with the server for transfer, and we name it buffered knowledge transfer. Each slice of buffered knowledge does not have to be generated from the same edge model. It relaxes the FL's synchronization requirements that the devices need to have the same model version (i.e. same edge-cloud communication frequency) and the same local epochs ($\forall i, j \in [K], E_d^i = E_d^j = E_d$). Moreover, due to the powerful computation resource in the cloud, we can have a larger number of cloud training epochs ($E_s \gg E_d$), and it can generate more knowledge-representative embeddings from enriched on-cloud features to aid the devices' models for fast training and better personalization.

Inference Strategy. During inference, there are two choices of strategies, i.e., the cloud-based and the edge-based ones. The cloud (edge) models use the transferred embeddings from the edge (cloud) and their own features to infer prediction for a given sample. Generally, the edge-based inference is more real-time and personalized while the cloud-based one is more generalized and robust. The inference strategies can be flexibly chosen according to the specific application scenarios.

3.2 Practical Add-ons

Two-stage strategy. We notice that in the early training phase when the models are less generalized, the knowledge of the logits is poor in representational power. As a result, the knowledge distillation may cause distortion in training. Thus, we only transfer the embeddings and use CE loss (i.e. $\alpha_s = \alpha_d = 0$) in the first few

SIGIR '23, July 23-27, 2023, Taipei, Taiwan.

training epochs, and additionally transfer the logits and use both KD and CE losses in later training phases.

Filtered Knowledge Transfer. We find that if a model predicts for some samples with wrong labels, the model is hardly generalized on these samples; therefore, taking the logits of these samples as knowledge will degrade performance in distillation [7]. Thus, we adopt filtered knowledge transfer that only uses the right logits for knowledge distillation.

Privacy Guarantee. The transfer of embeddings bears less privacy risk than directly transferring the device's raw private features [5, 8]. In our framework, the privacy guarantee can be further strengthened by applying differential privacy methods on the embeddings from the devices [1, 25].

4 EXPERIMENTS

In the experiments, we evaluate our framework on three datasets: CIFAR10, Avazu, and IndustryData. We introduce experimental settings in Section 4.1, present the results on CIFAR10 with synthetic feature split in Section 4.2, and illustrate the results on RS datasets (Avazu and IndustryData) in Section 4.3.

4.1 Experimental Settings

For fair comparisons, we consider three feature settings: "F" uses only federated features, "C2F" moves **c**entralized features **to** the edge as **f**ederated features, and "C&F" combines centralized features at the cloud and federated features at the edge. We note that "C2F" is not practical in real-world scenarios due to storage limitations and communication overhead. However, we include it for comprehensive evaluation. Our experiments evaluate personalization results on local test sets of edge devices, using three datasets. For the first 50 rounds of training, α_s and α_d are both set to 0, and then set to 1 for the latter 50 rounds. The transferred embeddings have a size of 128, and we employ an MLP with two layers as the downstream classifier.

Datasets. CIFAR10 [12] is an image dataset for classification, and we conduct synthetic feature split to generate federated and centralized features. Specifically, given a 32×32 image, we split it into a 10×32 image at the edge and a 22×32 image at the cloud. We deploy ResNet-8 at the edge and ResNet-56 at the cloud as the encoders. The number of devices is 20. Avazu [23] is a public dataset for click-through rate (CTR) prediction in real-world RS. We randomly sample 200,000 data samples and assign them to each device by device_ip, and there are 1769 devices. We set the sparse features, which are mostly about the item details (e.g. app_category), as the cloud-side centralized features and assume the dense features as the federated features. We use AutoInt [22] as the backbone encoders. IndustryData is a real-world industrial dataset for RS, extracted from system logs from Alipay APP. We randomly sample 500,000 data samples and assign them to 2000 devices. We allocate the user's sensitive features at the edge, like u_gender and u_occupation, and the historical interactive features as the centralized features at the cloud. AutoInt is used as the encoder, and we conduct experiments on both CTR and conversion rate (CVR) prediction tasks.

Baselines. We compare our framework with FL baselines, mainly FedAvg [18] (the general FL framework) and FedGKT [10] (the SOTA edge-cloud collaborative FL framework). For the experiments with CIFAR10, since the split features have the same feature spaces,

Table 1: Top-1 test accuracy (%) results on CIFAR10 under different device data heterogeneity and E_d . "homo"/"hetero": device model homogeneity/heterogeneity.

	Data heterogeneity	IID		NonIID	
Feature	Method/E _d	1	3	1	3
	FedAvg	57.36	59.75	52.36	56.96
F	FedGKT (homo)	66.78	67.90	38.88	44.59
	FedGKT (hetero)	51.57	54.34	23.36	26.20
C2F	FedAvg	61.81	66.32	57.09	62.69
	FedGKT (homo)	78.44	79.05	30.91	57.70
	FedGKT (hetero)	66.09	66.04	30.38	49.30
C&F	FedDF	31.9	49.14	28.86	41.36
	FedBE	38.28	51.55	31.73	43.09
	ECCT (homo)	81.89	82.46	63.06	62.61
	ECCT (hetero)	77.31	77.18	60.86	64.06

Table 2: Top-1 test accuracy (%) results on CIFAR10 under asynchronization.

Method	Fed	lAvg	ECCT		
Select ratio	1.0	0.5	1.0	0.5	
Syn.	62.69	55.78 11.0% ↓	67.81 -	69.28 2.1% ↑	
Asyn. version	60.51	53.29	67.56	64.42	
	3.5% ↓	15.0% ↓	0.4% ↓	5.3% ↓	
Asyn. epoch	59.55	50.93	66.92	64.65	
	5.0% ↓	18.8% ↓	1.3% ↓	4.9% ↓	
Asyn. both	58.47	51.49	67.31	64.81	
	6.7% ↓	17.9% ↓	0.7% ↓	4.6% ↓	

we also test distillation-based FL methods (FedDF [17] and FedBE [6]) that can utilize the centralized features via distillation.

4.2 Synthetic Feature Split

In this section, experiments are conducted on CIFAR10 with synthetic feature split. Table 1 shows the results under different device data heterogeneity and numbers of edge training epochs (E_d) using Dirichlet sampling [17] to assign devices with NonIID samples. Our method outperforms the baselines in both IID and NonIID settings. Moreover, our method shows a smaller decrease in performance compared with FedGKT when device model heterogeneity exists. Distillation (FedDF and FedBE) on the centralized features results in worse model performance, indicating that conventional FL methods cannot effectively utilize centralized features.

Table 2 presents the results under asynchronous settings. The degradation in performance is stronger in FedAvg than in ECCT, especially when combined with uniformly random partial device selection in each epoch of training.

Table 3 showcases the performance of different methods with various device numbers and selection ratios. Our ECCT outperforms FedAvg and FedGKT under all settings and is more tolerant with different participation of edge devices.

4.3 **Recommender Systems**

In this section, we implement the methods in RS, and both public (Avazu) and industrial datasets (IndustryData) are used.

Table 3: Top-1 test accuracy (%) results on CIFAR10 under different device numbers and device selection ratios.

ECCT	68.43	68.68	45.33	78.36	75.91	67.01
FedGKT	31.06	44.84	45.16	59.68	57.82	52.53
FedAvg	47.64	33.99	14.12	44.04	31.55	16.19
Select ratio	0.6	0.3	0.1	0.6	0.3	0.1
Device Num.		50			100	

Table 4: CTR prediction for Avazu.						
Feature	:	F	С	C&F		
Method	FedAvg	FedGKT	FedAvg	FedGKT	ECCT	
AUC MSE	0.5783 0.1370	0.5293 0.1488	0.6595 0.1320	0.5246 0.1559	0.6694 0.1373	

Table 5: CTR and CVR predictions for IndustryData.

Task	CTR		CVR		
Method/Metric	AUC	MSE	AUC	MSE	
FedAvg	0.6829	0.0879	0.8145	0.0199	
FedGKT	0.5409	0.1023	0.6068	0.0262	
ECCT	0.6885	0.0875	0.8754	0.0184	

Firstly, we conduct experiments on Avazu for CTR prediction tasks and demonstrate the results in Table 4. Our method is superior to FedAvg and FedGKT in terms of AUC, even moving all the centralized features to the edge ("C2F"). Secondly, we test the methods on IndustryData for CTR and CVR prediction tasks. The results in Table 5 illustrate that our edge-cloud collaborative framework performs better than the FL methods, especially in the CVR task. It shows that the framework has large potential in the industry, especially for online recommendations.

We notice that FedGKT has poor results in RS, and we find it reasonable. In FedGKT, the knowledge from the cloud to the edge is solely based on the logits. For binary classification tasks like predicting CTR and CVR, the logits contain no more information than the labels, so FedGKT fails to make an effective transfer from the cloud to the edge. However, our framework incorporates more knowledge in the transferred embeddings.

5 CONCLUSION

In this paper, we proposed an edge-cloud collaborative knowledge transfer framework to jointly utilize the centralized and federated features. The proposed method has several advantages over the previous federated learning methods: boosting edge personalization, enabling model heterogeneity, tolerating asynchronization, and relieving communication burdens between edge and cloud.

ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Project of China (2021ZD0110400), the National Natural Science Foundation of China (U19B2042), the Program of Zhejiang Province Science and Technology (2022C01044), The University Synergy Innovation Program of Anhui Province (GXXT-2021-004), Academy Of Social Governance Zhejiang University, Fundamental Research Funds for the Central Universities (226-2022-00064). Edge-cloud Collaborative Learning with Federated and Centralized Features

REFERENCES

- Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. 308–318.
- [2] Hédy Attouch, Jérôme Bolte, Patrick Redont, and Antoine Soubeyran. 2010. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Łojasiewicz inequality. *Mathematics of operations research* 35, 2 (2010), 438–457.
- [3] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. 2014. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming* 146, 1 (2014), 459–494.
- [4] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. Proceedings of Machine Learning and Systems 1 (2019), 374–388.
- [5] Fahao Chen, Peng Li, Toshiaki Miyazaki, and Celimuge Wu. 2021. Fedgraph: Federated graph learning with intelligent sampling. *IEEE Transactions on Parallel* and Distributed Systems 33, 8 (2021), 1775–1786.
- [6] Hong-You Chen and Wei-Lun Chao. 2020. FedBE: Making Bayesian Model Ensemble Applicable to Federated Learning. In International Conference on Learning Representations.
- [7] Sijie Cheng, Jingwen Wu, Yanghua Xiao, and Yang Liu. 2021. Fedgems: Federated learning of larger server models via selective knowledge fusion. arXiv preprint arXiv:2110.11027 (2021).
- [8] Xingbo Fu, Binchi Zhang, Yushun Dong, Chen Chen, and Jundong Li. 2022. Federated graph machine learning: A survey of concepts, techniques, and applications. arXiv preprint arXiv:2207.11812 (2022).
- [9] Yu Gong, Ziwen Jiang, Yufei Feng, Binbin Hu, Kaiqi Zhao, Qingwen Liu, and Wenwu Ou. 2020. EdgeRec: recommender system on edge in Mobile Taobao. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management. 2477-2484.
- [10] Chaoyang He, Murali Annavaram, and Salman Avestimehr. 2020. Group knowledge transfer: Federated learning of large cnns at the edge. Advances in Neural Information Processing Systems 33 (2020), 14068–14080.
- [11] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 2, 7 (2015).
- [12] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
- [13] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Process. Mag.* 37, 3 (2020), 50–60. https://doi.org/10.1109/MSP.2020.2975749
- [14] Zexi Li, Jiaxun Lu, Shuang Luo, Didi Zhu, Yunfeng Shao, Yinchuan Li, Zhimeng Zhang, Yongheng Wang, and Chao Wu. 2022. Towards Effective Clustered Federated Learning: A Peer-to-peer Framework with Adaptive Neighbor Matching. *IEEE Transactions on Big Data* (2022).
- [15] Zexi Li, Feng Mao, and Chao Wu. 2022. Can we share models if sharing data is not an option? Patterns 3, 11 (2022), 100603.

- [16] Shih-wei Liao, Tzu-Han Hung, Donald Nguyen, Chinyen Chou, Chiaheng Tu, and Hucheng Zhou. 2009. Machine learning-based prefetch optimization for data center applications. In Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis. 1–10.
- [17] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. 2020. Ensemble distillation for robust model fusion in federated learning. Advances in Neural Information Processing Systems 33 (2020), 2351–2363.
- [18] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273-1282.
- [19] Minh NH Nguyen, Huy Q Le, Shashi Raj Pandey, and Choong Seon Hong. 2022. CDKT-FL: Cross-Device Knowledge Transfer using Proxy Dataset in Federated Learning. arXiv preprint arXiv:2204.01542 (2022).
- [20] Xufeng Qian, Yue Xu, Fuyu Lv, Shengyu Zhang, Ziwen Jiang, Qingwen Liu, Xiaoyi Zeng, Tat-Seng Chua, and Fei Wu. 2022. Intelligent Request Strategy Design in Recommender System. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 3772–3782.
- [21] Saim Salman, Christopher Streiffer, Huan Chen, Theophilus Benson, and Asim Kadav. 2018. DeepConf: Automating data center network topologies management with machine learning. In Proceedings of the 2018 Workshop on Network Meets AI & ML. 8-14.
- [22] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via selfattentive neural networks. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management. 1161–1170.
- [23] Will Cukierski Steve Wang. 2014. Click-Through Rate Prediction. https://kaggle.com/competitions/avazu-ctr-prediction
 [24] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMa-
- [24] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, et al. 2021. A field guide to federated optimization. arXiv preprint arXiv:2107.06917 (2021).
- [25] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3454–3469.
- [26] Jiangchao Yao, Feng Wang, Xichen Ding, Shaohu Chen, Bo Han, Jingren Zhou, and Hongxia Yang. 2022. Device-cloud Collaborative Recommendation via Meta Controller. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 4353–4362.
- [27] Jiangchao Yao, Feng Wang, Kunyang Jia, Bo Han, Jingren Zhou, and Hongxia Yang. 2021. Device-cloud collaborative learning for recommendation. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 3865–3874.
- [28] Hong Zhang, Ji Liu, Juncheng Jia, Yang Zhou, Huaiyu Dai, and Dejing Dou. 2022. FedDUAP: Federated Learning with Dynamic Update and Adaptive Pruning Using Shared Data on the Server. arXiv preprint arXiv:2204.11536 (2022).