2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS) | 978-1-6654-7315-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICPADS56603.2022.00107

FedDyn: A dynamic and efficient federated distillation approach on Recommender System

Cheng Jin*, Xuandong Chen*, Yi Gu†, Qun Li
School of Computer Science & Technology
Nanjing University of Posts and Telecommunications, Nanjing, China

***Abstract*—Federated Learning (FL) is a popular distributed machine learning paradigm that enables devices to work together to train a centralized model without transmitting raw data. However, when the model becomes complex, mobile devices' communication overhead can be unacceptably large in traditional FL methods. To address this problem, Federated Distillation (FD) is proposed as a federated version of knowledge distillation. Most of the recent FD methods calculate the model output (logits) of each client as the local knowledge on a public proxy dataset and do distillation with the average of the clients' logits on the server side. Nevertheless, these FD methods are not robust and perform poorly in the non-IID (data is non-independent and non-identically distributed) scenario such as Federated Recommendation (FR). In order to eliminate the non-IID problem and apply FD in FR, we proposed a novel method named *FedDyn* to construct a proxy dataset and extract local knowledge dynamically in this paper. In this method, we replaced the average strategy with *focus distillation* to strengthen reliable knowledge, which solved the non-IID problem that the local model has biased knowledge. The average strategy is a dilution and perturbation of knowledge since it treats reliable and unreliable knowledge equally important. In addition, to prevent inference of private user information from local knowledge, we used a method like local differential privacy techniques to protect this knowledge on the client side. The experimental results showed that our method has a faster convergence speed and lower communication overhead than the baselines on three datasets, including MovieLens-100K, MovieLens-1M and Pinterest.**

***Index Terms*—Distributed Machine Learning, Federated Learning, Recommender System, Knowledge Distillation**

I. Introduction

With the proliferation of the Internet, information overload has gradually become a serious problem. The huge amount of items may overwhelm people and make it difficult to find what they like. As an effective tool to solve information overload, Recommender System (RS) can help users discover the information they are interested in and help service providers increase traffic and revenue. It has been widely used on platforms such as e-commerce and social media. Research on RS can date back to the 1990s [1], and many works based on collaborative filtering have been developed [2]. After that, Matrix Factorization (MF) [3]–[5] became the mainstream recommender model for a long time (from 2008 until 2016). However, due to the fact that the factorization model is inherently linear, its performance is lacking in handling large-scale and complex data, such as complex user interactions and the item side that may contain complex semantic information. Later, with the rapid development of deep learning, many works emerged on developing neural network approaches to recommender systems [6].

In recent years, due to the improvement of user privacy awareness in society and the implementation of data privacy laws, such as the General Data Protection Regulations (GDPR, effective May 2018), the California Privacy Rights Act (CPRA, effective January 2021) and the China Data Security Law (CDSL, Effective September 2021), it gradually became impossible for the central server to obtain user data training directly. Private machine learning has attracted more and more attention from academia and industry (such as Google and WeBank) [7]. Besides, the requirements of training complex RS models for millions of users and items on a central server raises scalability issues, requires significant computing and storage resources, and high server maintenance costs for companies [8]. Therefore, Federated Learning (FL) [9] has been proposed to tackle these concerns by distributed training. FL has proved suitable for various application scenarios, from mobile devices with limited resources to resource-rich institutions. During the training process, the user's raw data is kept locally by the user all the time. The server and the user carry out model training and parameter updates by sharing encrypted or non-containing intermediate parameters to build an effective machine to protect user privacy.

However, there are still two main challenges left unsolved. On the one hand, frequently exchanging model parameters over the training process leads to an excessive communication burden. On the other hand, homogeneous models among clients conflict with client heterogeneity in data distribution and system configuration. These defects pose a severe threat to the effectiveness of an FL method and can easily result in drastic performance drops and hinder the actual deployment of FL [10]. Motivated by these defects, Federated Distillation (FD) is thus proposed as a federated version of Knowledge Distillation [11], [12], in which logits (also called knowledge) become the intermediate parameters exchanged between clients and the server. Most existing FD methods use a public proxy dataset to calculate each client's logits (the local knowledge) and do distillation on the server side with the average of the clients' logits. Because FD transfers far fewer parameters than traditional FL methods and knowledge can be decoupled from the model structure, FD can maintain a low communication overhead while allowing personalized models to be designed

* Equal contribution.
† Corresponding author, email: gy@njupt.edu.cn

978-1-6654-7315-6/23/$31.00 ©2023 IEEE
DOI 10.1109/ICPADS56603.2022.00107

for individual clients.

Because FD optimizes the overhead of a single communication, it is not guaranteed to converge in the non-IID scenario. Therefore, most of the recent FD methods cannot be used in FR since it is a typical non-IID scenario. As demonstrated in the abstract, taking the average in such a case is equivalent to adding noise to the knowledge which will lead to unstable server-side optimization under biased knowledge supervision and eventually deviate from the global optimum. To tackle this, we proposed a novel FD method called *FedDyn* to extract local knowledge dynamically, focusing on each client's knowledge. To avoid leaking user information from a single client output, we also tested the performance of an integration strategy that combines local noise.

The main contributions of this work include:

- We proposed a federated distillation recommender system (*FedDyn*), which can be trained much faster and requires fewer communication rounds than baseline.
- We conduct thorough experiments on three real-world datasets to verify the effectiveness and efficiency of our approach.

II. Related Work

A. Recommender System

In the last decades, Collaborative Filtering (CF) has been far the most successful, well-known, and widely investigated recommender system approach in academia and industry [13]. The core idea of collaborative filtering-based recommendation algorithms is to recommend items to users that have been interacted with by groups of users with similar history preferences. This approach allows delivering personalized recommendations to users based only on their past ratings on items. o do so, CF should predict the rating users would assign to unseen items, which can be addressed as a matrix completion problem [14]. In other words, the rating matrix, *i.e.*, a user-item matrix partially fulfilled with ratings, has its empty cells completed with rating predictions [15]. There is vast literature on this issue in Recommender Systems, in which most of the proposals are based on matrix factorization methods [16]. Matrix factorization (MF) is one of the most popular collaborative filtering recommendation algorithms, which takes a high-dimensional (user, item) rating matrix as input, outputs a low-dimensional user feature matrix and a low-dimensional item feature matrix, and calculates the user rating matrix of the item by the inner product of the user feature matrix and the item feature matrix.

With the popularity of deep learning techniques, deep learning has become an important technique in the field of recommendation systems. Compared with traditional recommendation algorithms, deep learning-based recommendation algorithms are more expressive and better able to explore the potential features of data and obtain deep user and item descriptions. In this context, Neural Collaborative Filtering (NCF) emerged as one of the most popular methods [17]. Roughly speaking, this is a generalization of the Matrix Factorization framework, which estimates user/item interactions throughout latent factors. NCF proposes to extend the linear model by using neural network layers, *i.e.*, complex non-linear transformations by avoiding the limitations imposed by the classic matrix factorization methods [16].

B. Federated Learning

To improve scalability and overcome the need for central data collection, a new method for training machine learning models in a collaborative and distributed way has emerged, named Federated Learning (FL) [9], [18]. In order to achieve secure, efficient and accurate federated learning, data heterogeneity is an urgent problem to be solved, which means a device's local data cannot be regarded as samples drawn from the overall distribution, and the data available locally fail to represent the overall distribution [19]. *Zhu et al.* [20] have compiled a comprehensive survey on the impact of non-IID data on federated learning, also reviewing the current research on handling these challenges. To improve training on non-IID data, *Li et al.* [21] proposed the FedProx framework, which uses the l2 norm of the difference between the client and server weights as the regular term for client training, *Zhao et al.* [22] created a small subset of data which is globally shared between all the edge devices and *Li et al.* [23] used local batch normalization to alleviate the feature shift before averaging models.

C. Federated Recommender System

Despite FL being a promising technique, its application on RS has not received much attention [8]. *Ammad et al.* [24] proposed a FedCF algorithm based on implicit feedback. Although their system only requires each user's update of the item profile, the raw gradients information can result in information leakage [25]. *Chai et al.* [25] proposed FedMF, which is an extension of the system in [24] by integrating a HE scheme. *Lin et al.* [26] explored a federated MF system in the context of explicit feedback. Their system, FedRec, randomly samples unrated items with a parameter $\rho \in 0, 1, 2, 3$ and assigns them a virtual score to hide a user's behavior and enhance privacy of the participants [27]. *Perifanis and Efraimidis* [27] presented a federated version of Neural Collaborative Filtering (NCF) approach and integrated a privacy-preserving enhancement with a secure aggregation scheme that satisfies the security requirements against an honest-but-curious (HBC) entity, without affecting the quality of the original model. *Muhammad et al.* [8] focused on the convergence rate of the federated recommender system and proposed a clustering method to accelerate convergence. However, as a trade-off, clients must upload their user features to the server for clustering, which obviously does harm user privacy.

D. Federated Knowledge Distillation

Knowledge distillation for neural networks (KD) is a teacher-student learning paradigm that transfers the teacher model's knowledge to the student model through distillation, which is first introduced in [11], [28]. Typical federated knowledge distillation (FD) methods [29]–[31] exchange

model outputs instead of model parameters among clients and the server. The server calculates the average of clients' logits as an aggregated representation of client knowledge and guides clients to converge towards global generalization. These methods, however, require a proxy dataset without exception, which is often not available during the FD process [10]. To tackle this, recent work focus on exchanging additional information, such as global models [32], generator [33], or extracted features [34].

III. Methodology

This section introduces our FedDyn method for fast and efficient recommendations. We first introduce the problem formulation and the local recommendation model. Then we introduce the framework and flow of our FedDyn method. After that, we describe the specific operations of each stage separately.

A. Problem Formulation

The recommendation algorithms aim to select items and present them to users based on their past behaviors. In a typical scenario, a dataset has N clients and M items. We denote $U = \{u_1, u_2, \ldots, u_n\}$ be the set of all clients and $I = \{i_1, i_2, \ldots, i_m\}$ be the set of all items. In each client $u \in U$, we use W^u to denote the full set of parameters of the local model, and in t-th local training round, we use W_t^u to denote the local model's parameters. Also, we use W_r to represent the server model parameters in the r-th round. For Federated Distillation, we denote E as the distillation temperature and $\mathbf{d}_u$ as the user-item pairs selected from the client u. Then, let $f(d; W)$ be the logits of a model with W as its parameters and d ad its input.

The main goal is to efficiently train an accurate recommendation model with the FD method. Besides, it should be noted that since all the FL algorithms mentioned in this paper select part of clients for training in each round, the number of iterations R on the server side and the number of local training rounds T on the client side is different and grow asynchronously.

Detailed notations and descriptions are given in TABLEI.

TABLE I: MAIN NOTATIONS WITH DESCRIPTIONS.

Notation	Description
N	The number of clients
M	The number of items
u	A client's (user's) ID
i	A item's ID
v_u^U	The feature vector of user u
v_i^I	The feature vector of item i
T	The number of local training rounds
R	The number of server iterating rounds
W_t^u	The u-th client's model parameters at t-th local round
W_r	The server's model parameters at r-th iterating round
E	The distillation temperature
$\mathbf{d}_u$	The extracted features from client u
$f(d; W)$	The logits of a model with W parameters and d input

B. Local Recommendation Model

To demonstrate the generality of our framework, we choose the NeuMF model introduced in [16] as the recommender to train on the client side. The NeuMF model is a neural network based on matrix factorization and multi-layer perceptron that uses user-id and item-id as the inputs and the probability of user-item interaction $\hat{y}_{ui}$ as the output. The whole process can refer in Figure 1.

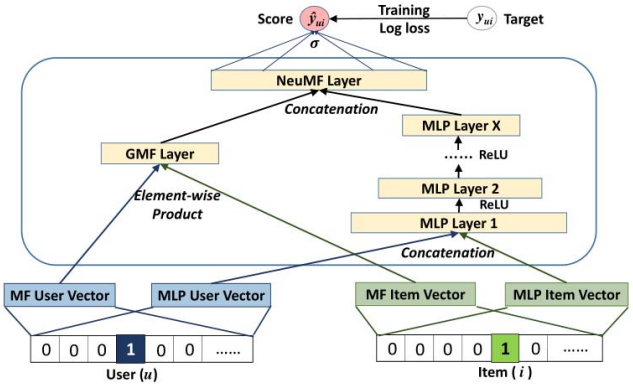


Fig. 1: Neural matrix factorization model [16]

The original input is user-id U_i and item-id I_i. After one-hot encoding, we get two feature vectors v_u^U and v_i^I that describe user u and item i separately. Both vectors can be customized to support various modeling of users and items. Similar to the NCF considerations, in order to focus on the testing and improvement of the FD method, we use only the identity of a user and an item as the input feature.

A fully-connected embedding layer projects the v_u^U and v_i^I vectors to their latent vector representations. Then, with denoting the user latent vector p_{uk} and the item latent vector q_{ik} (here k denotes the embedding size), GMF layer conducts an element-wise product between p_{uk} and q_{ik} while MLP layers use concatenation of p_{uk} and q_{ik} as input. Finally, the NeuMF layer is a fully connected layer using a concatenation of the output of the GMF layer and MLP layers as an input. The whole model for combining GMF with a one-layer MLP can be formulated as

$$\hat{y}_{ui} = \sigma\left(\mathbf{h}^T a\left(\mathbf{p}_{uk} \odot \mathbf{q}_{ik} + \mathbf{W}\begin{bmatrix}\mathbf{p}_{uk}\\ \mathbf{q}_{ik}\end{bmatrix} + \mathbf{b}\right)\right) \quad (1)$$

C. System Overview

As illustrated in Figure 2, our system is a client-server architecture FD method. Each user participating in the training is treated as a separate client. In other words, each client has a record of interaction with only one user and a set of items. In order to reduce the communication overhead, we adopt the technique of federated distillation so that the client only needs to upload local knowledge (logits) to the server when training. Since the motivation of this paper comes from the knowledge dilution caused by average strategy and our solution is based

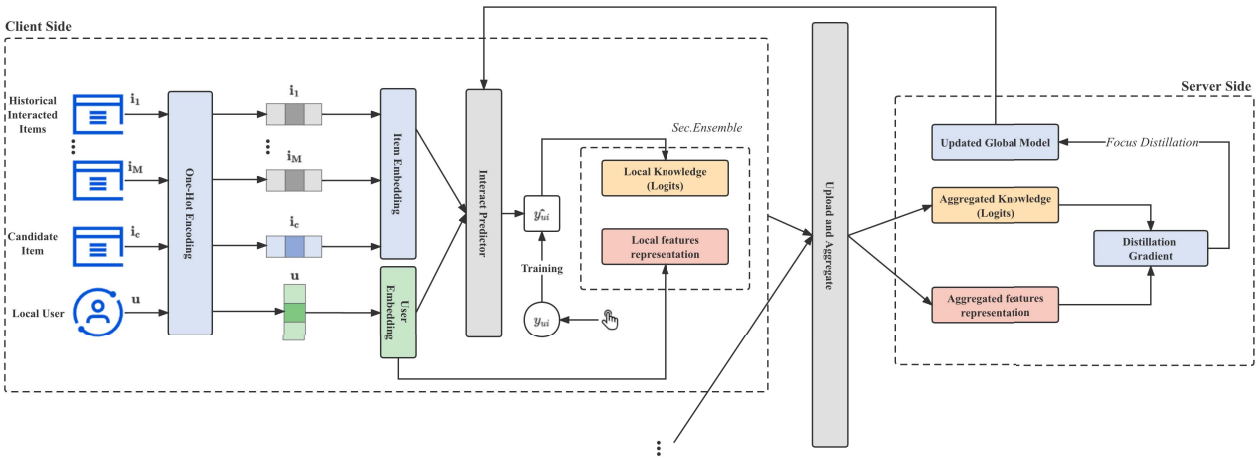


Fig. 2: FedDyn architecture

on a dynamic proxy data construction focusing on knowledge extraction, we call our system *FedDyn*.

The overall process is as follows:

1. The server and all clients initialize their model parameters;
2. The server selects several clients and sends a signal asking them to train with local data;
3. The client receiving the signal performs multiple rounds of training using local data
4. After training, the client executes the *Sec. Ensemble* and uploads local knowledge generated by the procedure.
5. The server receives the local knowledge, updates the global model by distilling local knowledge and sends the new model back to the client;
6. Repeat steps 2-5 until the model converges.

client train with private

Algorithm 1 Overview of FedDyn System

procedure SERVER
  $w_0, w_0^{u_0}, w_0^{u_1}, \ldots, w_0^{u_n} \leftarrow$ initial model parameters
  for each communication round $t = 1, \ldots, R$ **do**
    $U_r \leftarrow$ sampled subset of the C
    for each client $u_k \in U_r$ **in parallel do**
      $\hat{w}_r^{u_k} \leftarrow$ Client-Train(u_k, w_{r-1})
      $d_r^{u_k}, l_r^{u_k} \leftarrow$ Client-SecEnsemble(u_k)
      Upload $d_r^{u_k}, l_r^{u_k}$ to the server
    end for
    $d_r, l_r \leftarrow$ Aggregation$(\{d_r^{u_k}\}_{u_k \in U_r}, \{l_r^{u_k}\}_{u_k \in U_r})$
    $w_t \leftarrow$ Focus Distillation(d_r, l_r, w_{r-1})
  end for
  return w_T
end procedure

1, Local Train
2, 生成数据
3,
4, Applied KD

D. Sec.Ensemble

After the clients finish training the local model, we must execute a proxy data construction for knowledge extraction and distillation. Our motivation for designing the proxy data construction method is that randomly constructed data cannot guarantee the final convergence of the system, and the training efficiency is very low. Thus, to refine the local knowledge, we propose a new knowledge extraction method called *Sec.Ensemble*. In each iteration, clients evaluate items with their local models and participate in the generation of the proxy data dynamically, passing local features and logits extracted by the model to the server. To avoid information leakage, we refer to the LDP method and let the client add some fake ids when generating local features. Thus, we call this procedure *Sec.Ensemble* and show the algorithm 2 below.

Algorithm 2 Function of Sec.Ensemble

function SEC.ENSEMBLE(u_k)
  for i in $\{1, \ldots, I\}$ **do**
    $\hat{y}_{ui}, l_{ui} \leftarrow$ predict with model inu_k
  end for
  Select top-K user-item pairs in interaction probability
  Randomly select several user-item pairs that are not in top-K pairs
  Merge two parts of data and send them and their logits to server
end function

In this procedure, we first evaluate all item-ids with the user-id belonging to the client u_k and select K items with the largest interaction probability. Meanwhile, we generate fake data with a random user-id and K random item ids. After calculating the features of the selected user-item pairs with fake data, the client uploads the features with those logits $l_r^{u_k}$ to the server. After the server receives all the data uploaded

通过merge的方式生成pesudo data，然后基于这些数据distill出logits

by the selected client, it performs integration. It shuffles to complete the proxy data construction for distillation, where the proxy data comprises the extracted features d_r and their logits l_r.

E. Focus Distillation

logits avg

After receiving the data, the server side must perform distillation to aggregate the local knowledge from clients. Our distillation method is different from other federated distillation methods, which use the averaging of logits commonly. We use logits generated by the client model with the same user as input since we think one client can only extract the knowledge from the corresponding user in FR. One iteration j of the procedure can be formulated as

$$\mathbf{w}_{r,j} \leftarrow \mathbf{w}_{r,j-1} - \eta \frac{\partial \mathrm{KL}\left(\sigma\left(\frac{1}{E}\right), \sigma\left(\frac{f(\mathbf{w}_{r,j-1}, \mathbf{d})}{E}\right)\right)}{\partial \mathbf{w}_{r,j-1}} \quad (2)$$

Here KL stands for Kullback–Leibler divergence, σ is the softmax function, and η is the stepsize.

IV. Evaluation on the Common Federated Learning Settings

In this section, we demonstrate the efficiency and effectiveness of our FedDyn. We conduct several experiments to answer the following research questions:

- **RQ1:** How does our method perform compared with the baselines?
- **RQ2:** Does the average strategy work in FR?
- **RQ3:** Are the communication overhead significantly reduced compared with the baselines?
- **RQ4:** Which hyperparameters impact system performance?

A. Dataset and Experimental Settings

1) Dataset: We conduct thorough experiments on three public datasets, i.e., MovieLens 100K[1], MovieLens 1M[2] and Pinterest[3]. The characteristics of the two datasets are summarized in TableII

TABLE II: Statistics of the evaluation datasets.

Dataset	#Interaction	#Item	#User	Sparsity
MovieLens 100K	100,000	1,682	943	93.7%
MovieLens 1M	1,000,209	3,706	6,040	95.53%
Pinterest	1,500,809	9,916	55,187	97.96%

1) **MovieLens** The first two datasets are movie ratings, where each user has interacted with at least 20 items. Because the interaction is explicit feedback data, we transformed it into implicit data, using 0 and 1 to indicate whether the user has rated the item.
2) **Pinterest** This implicit feedback dataset is constructed by [35] for evaluating content-based image recommendation. The original data is very large but highly sparse. For example, over 20% users have only one pin, making it difficult to evaluate collaborative filtering algorithms. As such, we filtered the dataset in the same way as the MovieLens data that retained only users with at least 20 interactions (pins).

[1]https://grouplens.org/datasets/movielens/100K/
[2]https://grouplens.org/datasets/movielens/1m/
[3]https://sites.google.com/site/xueatalphabeta/academic-projects

2) Evaluation Protocol: For recommendation performance evaluation, we adopted the *leave-one-out* proctocol, which has been widely used in literature [8], [16], [27]. We held out their latest interaction as the test set for each user and utilized the remaining data for training. For quick evaluation, we followed the common strategy that randomly samples 100 items that are not interacted with by the user, ranking the test item among the 100 items [16]. Although this kind of evaluation may falsely produce better results, our goal is not to present a SOTA system, but to fairly compare the federated variant against the centralized one [16], on the same settings [27]. The performance of a ranked list is judged by Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG). Without special mention, we truncated the ranked list at 10 for both metrics. As such, the HR intuitively measures whether the test item is present on the top-10 list, and the NDCG accounts for the position of the hit by assigning higher scores to hits at the top ranks [16], [27], [36]. We calculated both metrics for each test user and reported the average score.

3) Baseline: FedDyn is designed for effective FD training in FR, considering the accuracy of the global model on the test dataset. Thus, we omitted to compare methods designed for robustness and communication efficiency, including 1) FedAvg [9], 2) FedProx [21], and 3) FedDF [37].

B. Performance Comparison (RQ1)

We must make an upfront statement that we have no intention of getting a SOTA model in recommendation performance. This experiment aims twofold: (i) to compare the impact of different FL algorithms on performance and (ii) to demonstrate the generality and robustness of our approach. The experimental results of all these methods are shown in Table III and Picture 3.

We have several observations from Table III. First, by comparing our FedDyn with baselines, our method achieves comparable performance. Therefore, it validates that our method can train accurate recommendation models and protect user privacy. Second, our method performs better than FedAvg. This is because FedAvg cannot tackle the non-IID problem in FR. Finally, FedDF performs the worst. This is probably because user behaviors are non-IID, which may make it difficult to construct an effective proxy dataset to achieve good results.

C. Average Strategy Analysis (RQ2)

To demonstrate the superiority of our dataset construction and distillation method, we compare *Sec.Ensemble* with generator (Gen), which generates data representing the global distribution and compares Focus Distillation with ensemble distillation which uses the average logits to distill (AvgLogits).

TABLE III: Recommendation Performance Overview

		FedDyn	FedAvg	FedProx	FedDF
ML100K	HR@10	0.585	0.544	0.557	0.361
	NDCG@10	0.367	0.346	0.359	0.231
ML1M	HR@10	0.552	0.547	0.556	0.345
	NDCG@10	0.313	0.297	0.301	0.232
Pinterest	HR@10	0.343	0.338	0.330	0.204
	NDCG@10	0.194	0.179	0.174	0.140

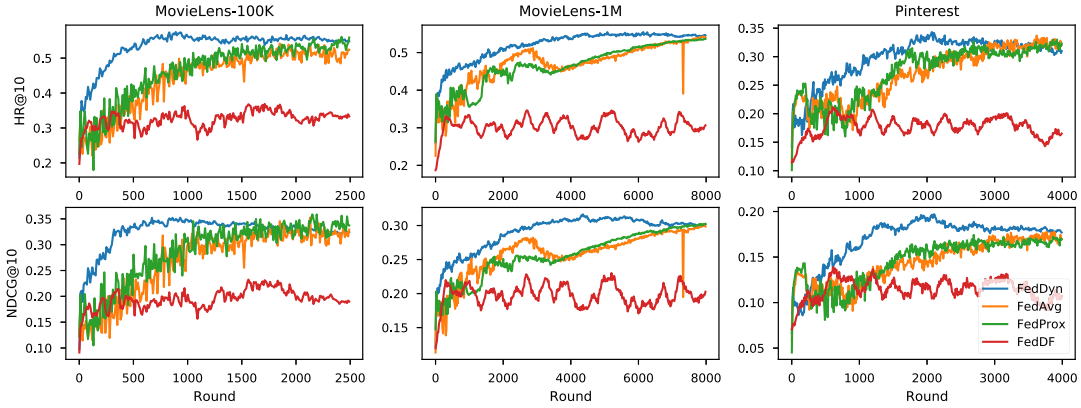


Fig. 3: Convergence speed between FedDyn and baselines on the evaluation datasets showing NDCG@10 and HR@10

The result of this experiment is presented in Figure 4. Obviously, the original FedDF method converged at a very low level and when we turn to *Sec.Ensemble* as proxy data construction method, FedDF converged the same as other federated learning methods while convergence speed is basically the same as FedAvg. After that, we changed the distillation method of FedDF (which actually became FedDyn), and performed much fast than other federated learning methods.

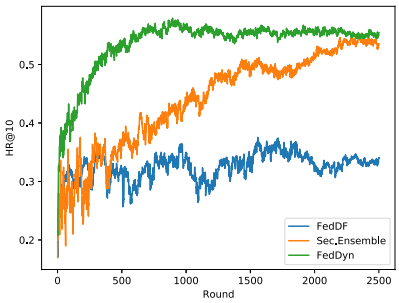


Fig. 4: Impact on *Sec.Ensemble* and *Focus Distillation*

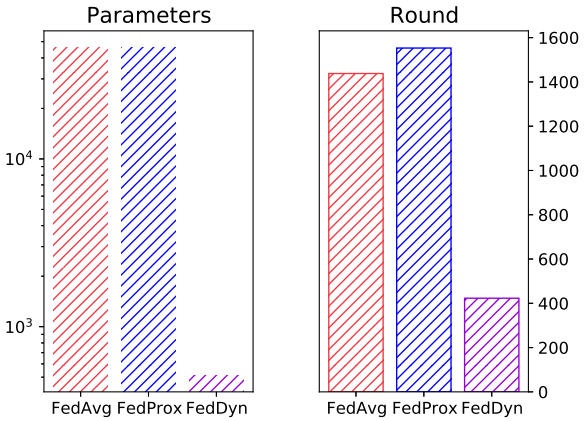
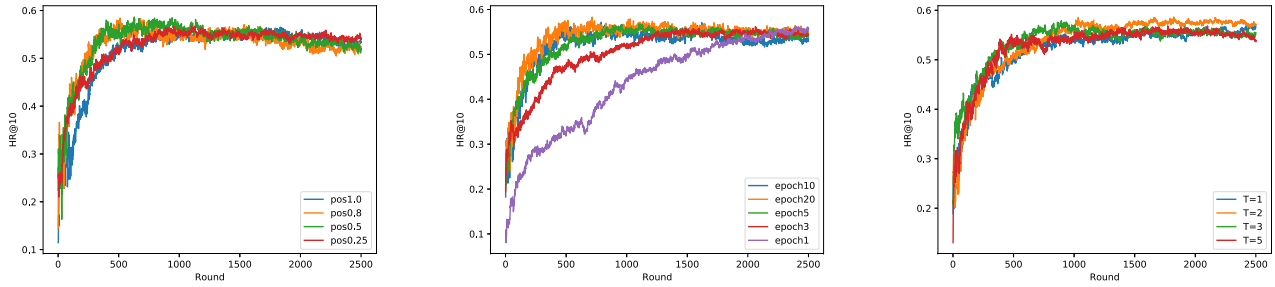


Fig. 5: The communication cost of FL methods on ML-100K.

D. *Efficiency Analysis (RQ3)*

To validate our claim of improved training efficiency, we compare the convergence between FedDyn and baselines. The results are shown in Figure 3 and Figure 5. Four methods start from the same HR@10 and NDCG@10, and FedDyn is faster than those of FedDF and FedProx. This is because FedAvg cannot handle the non-IID problem, while FedProx can guarantee convergence in non-IID scenarios with increasing training rounds as a trade-off. The FedDF method was performed with proxy data that conforms to global distribution but converges at a low position. Besides, FedDyn converges much more quickly than others and settles to within 10% of its best values in all evaluation datasets. This means that clients of FedAvg will have to endure 70% more communication rounds and expend 70% more local training effort to enjoy the same performance as the clients of FedDyn. Naturally, this significant difference in convergence speed can be attributed to



(a) Impact on positive ratio (b) Impact on local epochs (c) Impact on distillation temperature

Fig. 6: Key Hyper Parameters Impact

the impact of *Sec.Ensemble* and *Focus Distillation* in FedDyn.

E. Understading FedDyn (RQ4)

In this subsection, we compared three key hyperparameters: *Negative Ratio.*, *Distill Temperature.* and *Local Epochs*. The results are illustrated in Figure 6 and show that the distillation temperature and the proportion of positive and negative examples during data construction have little effect on the convergence rate. While local epochs have a significant impact, consistent with the FedAvg experiment.

V. Conclusion

In this paper, we proposed FedDyn, an efficient federated method for recommendations using federated distillation. FedDyn evaluates clients' local knowledge in each iteration and dynamically constructs proxy data based on the evaluation. Thus, local knowledge can be extracted properly. In addition, we replaced the average strategy with focus distillation, which is more effective and reliable in FR. Experiments on three real-world datasets validate that our method can reduce communication overhead effectively without compromising the model's recommendation accuracy. Future work will consider new approaches to make the data construction process even more efficient and safe. We will also consider more sophisticated models that improve the accuracy of the underlying recommendation model by extending its federated version.

References

[1] D. E. Goldberg, D. A. Nichols, B. M. Oki, and D. B. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of The ACM*, 1992.

[2] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, 2005.

[3] Y. Koren, R. M. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer*, 2009.

[4] A. Mnih and R. Salakhutdinov, "Probabilistic matrix factorization," *neural information processing systems*, 2007.

[5] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," *knowledge discovery and data mining*, 2008.

[6] L. Wu, X. He, X. Wang, K. Zhang, and M. Wang, "A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation," 2022.

[7] F. Liang, W. Pan, and Z. Ming, "Fedrec++: Lossless federated recommendation with explicit feedback," *national conference on artificial intelligence*, 2021.

[8] K. Muhammad, Q. Wang, D. O'Reilly-Morgan, E. Tragos, B. Smyth, N. Hurley, J. Geraci, and A. Lawlor, "Fedfast: Going beyond average for faster training of federated recommender systems," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1234–1242.

[9] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[10] Z. Wu, S. Sun, Y. Wang, M. Liu, and Q. Liu, "Exploring the distributed knowledge congruence in proxy-data-free federated distillation," *arXiv preprint arXiv:2204.07028*, 2022.

[11] G. Hinton, O. Vinyals, J. Dean *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[12] R. Anil, G. Pereyra, A. Passos, R. Ormandi, G. E. Dahl, and G. E. Hinton, "Large scale distributed neural network training through online distillation," *arXiv preprint arXiv:1804.03235*, 2018.

[13] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, pp. 1–45, 2014.

[14] J. Fan and J. Cheng, "Matrix completion by deep matrix factorization," *Neural Networks*, vol. 98, pp. 34–41, 2018.

[15] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE transactions on knowledge and data engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[16] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2017, pp. 355–364.

[17] M. F. Dacrema, S. Boglio, P. Cremonesi, and D. Jannach, "A troubling analysis of reproducibility and progress in recommender systems research," *ACM Transactions on Information Systems (TOIS)*, vol. 39, no. 2, pp. 1–49, 2021.

[18] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[19] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv: Machine Learning*, 2019.

[20] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-iid data: A survey," *Neurocomputing*, 2021.

[21] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *arXiv: Learning*, 2018.

[22] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data." *arXiv: Learning*, 2018.

[23] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "Fedbn: Federated learning on non-iid features via local batch normalization." *arXiv: Learning*, 2021.

[24] M. A. ud din, E. Ivannikova, S. A. Khan, W. Oyomno, Q. Fu, K. E. Tan, and A. Flanagan, "Federated collaborative filtering for privacy-preserving personalized recommendation system," *arXiv: Information Retrieval*, 2019.
[25] D. Chai, L. Wang, K. Chen, and Q. Yang, "Secure federated matrix factorization," *arXiv: Cryptography and Security*, 2019.
[26] G. Lin, F. Liang, W. Pan, and Z. Ming, "Fedrec: Federated recommendation with explicit feedback," *IEEE Intelligent Systems*, 2021.
[27] V. Perifanis and P. S. Efraimidis, "Federated neural collaborative filtering," *Knowledge-Based Systems*, vol. 242, p. 108441, 2022.
[28] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 535–541.
[29] S. Itahara, T. Nishio, Y. Koda, M. Morikura, and K. Yamamoto, "Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data," *IEEE Transactions on Mobile Computing*, 2020.
[30] D. Li and J. Wang, "Fedmd: Heterogenous federated learning via model distillation," *arXiv: Learning*, 2019.
[31] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data." *arXiv: Learning*, 2018.
[32] G. Lee, Y. Shin, M. Jeong, and S.-Y. Yun, "Preservation of the global knowledge by not-true self knowledge distillation in federated learning." *arXiv: Learning*, 2021.
[33] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," *arXiv: Learning*, 2021.
[34] C. He, M. Annavaram, and A. S. Avestimehr, "Group knowledge transfer: Federated learning of large cnns at the edge," *neural information processing systems*, 2020.
[35] X. Geng, H. Zhang, J. Bian, and T.-S. Chua, "Learning image and user features for recommendation in social networks," *international conference on computer vision*, 2015.
[36] X. He, T. Chen, M.-Y. Kan, and X. Chen, "Trirank: Review-aware explainable recommendation by modeling aspects," *conference on information and knowledge management*, 2015.
[37] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," *neural information processing systems*, 2020.