Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.061166

ARTICLE

# Integration of Federated Learning and Graph Convolutional Networks for Movie Recommendation Systems

**Sony Peng**[1], **Sophort Siet**[1], **Ilkhomjon Sadriddinov**[1], **Dae-Young Kim**[2,*], **Kyuwon Park**[3,*] and **Doo-Soon Park**[2]

[1]Department of Software Convergence, Soonchunhyang University, Asan, 31538, Republic of Korea
[2]Department of Computer Software and Engineering, Soonchunhyang University, Asan, 31538, Republic of Korea
[3]AI·SW Education Institute, Soonchunhyang University, Asan, 31538, Republic of Korea

*Corresponding Authors: Dae-Young Kim. Email: dyoung.kim@sch.ac.kr; Kyuwon Park. Email: q1park@sch.ac.kr

Received: 18 November 2024; Accepted: 01 March 2025; Published: 16 April 2025

**ABSTRACT:** Recommendation systems (RSs) are crucial in personalizing user experiences in digital environments by suggesting relevant content or items. Collaborative filtering (CF) is a widely used personalization technique that leverages user-item interactions to generate recommendations. However, it struggles with challenges like the cold-start problem, scalability issues, and data sparsity. To address these limitations, we develop a Graph Convolutional Networks (GCNs) model that captures the complex network of interactions between users and items, identifying subtle patterns that traditional methods may overlook. We integrate this GCNs model into a federated learning (FL) framework, enabling the model to learn from decentralized datasets. This not only significantly enhances user privacy— a significant improvement over conventional models but also reassures users about the safety of their data. Additionally, by securely incorporating demographic information, our approach further personalizes recommendations and mitigates the cold-start issue without compromising user data. We validate our RSs model using the open MovieLens dataset and evaluate its performance across six key metrics: Precision, Recall, Area Under the Receiver Operating Characteristic Curve (ROC-AUC), F1 Score, Normalized Discounted Cumulative Gain (NDCG), and Mean Reciprocal Rank (MRR). The experimental results demonstrate significant enhancements in recommendation quality, underscoring that combining GCNs with CF in a federated setting provides a transformative solution for advanced recommendation systems.

**KEYWORDS:** Recommendation systems; collaborative filtering; graph convolutional networks; federated learning framework

## 1 Introduction

The rise of the internet has resulted in an increase of user-generated data across many different platforms, including e-commerce sites and other social media networks. The flood of data presents both obstacles and potential for the creation of more customized and efficient systems, especially through recommendation algorithms [1,2]. CF is an often-used technique in recommendation systems (RSs), employing user behavior to suggest favored products or items [3,4]. Nevertheless, conventional CF approaches frequently encounter challenges like data sparsity, cold-start difficulties, and scalability limitations.

Moreover, the majority of personalized RSs gather user data to improve the precision of suggestions. Reference [5] presented recommendation systems that adaptively acquired user preferences, necessitating the gathering of user input to effectively suggest movies.

To further understand user information, references [6,7] explored a recommendation system utilizing community detection methods from various perspectives, with reference [6] designing a movie recommendation system based on personal information and rated movies using the k-clique method and NDCG. Additionally, reference [8] deployed a model that combined deep learning to enhance the recommendation system. Researchers have also been exploring graph algorithms, including GCNs, which are gaining attention due to their ability to handle complex user-item interaction graphs.

GCNs have become a powerful method for handling graph-structured data, offering a new perspective on CF methodologies [9]. GCNs model users and items as nodes in a graph, with edges indicating interactions/connections or similarities, thus capturing complex interaction patterns that other traditional methods, such as matrix factorization or nearest neighbor, often miss. GCNs excel by propagating information across the graph structure, thus improving node representations with contextual information from adjacent nodes. However, integrating GCNs into real applications poses significant privacy and data ownership challenges. Traditional approaches require consolidating all data in a single repository, increasingly at odds with contemporary data protection regulations such as GDPR (General Data Protection Regulation) in Europe, CCPA (California Consumer Privacy Act) in California, and global consumer privacy expectations [10]. This situation sets the stage for FL, an innovative machine learning paradigm that trains models across multiple decentralized devices or servers, which retain local data samples without transferring them [11].

This research proposes an architecture that combines GCNs and FL for CF in RSs. The contribution of our recommendation is as follows:

- Enhance Recommendation Quality: By utilizing the expressive potential of GCNs to comprehend complex user-item interaction patterns more effectively than traditional CF methods and by integrating user demographic and movie features to provide more personalized recommendations, particularly for new users through a distributed learning approach that minimizes data transfer and optimizes local computation. This research is essential for improving recommendation systems and illustrating a practical use of GCNs inside a privacy-preserving distributed learning framework. Furthermore, we want to address the cold-start problem inherent in conventional recommendation models.
- Personalization: By integrating user demographics and item content, we want to provide personalized recommendations and address the cold-start challenges for users and items. Consequently, we aim to investigate how FL and Graph Convolutional Networks (GCNs) may preserve or improve the predictive accuracy of centralized models while ensuring user privacy, thus defining a new benchmark for future research in privacy constraint-based recommendation systems based on graph-structured data.
- Enhance Privacy: By ensuring that personal interaction data remains within the user's device, thus complying with stringent privacy standards.

The structure of this article is prepared as follows. The paper starts with the overall recommendations and introduction of the fundamentals of each component for study in Sections 1 and 2. Section 3 introduces the methodology and details of the study. Section 4 experiments with the process described in Section 3. The details of the baseline are also presented in Section 4. Sections 5 and 6 present the ablation study and the conclusion, respectively.

## 2 Related Work

### *2.1 Collaborative Filtering Techniques*

CF is fundamental in RSs and is traditionally divided into memory-based and model-based approaches. Memory-based CF predicts user preferences using user-to-user or item-to-item correlations from the interaction matrix. Although these methods are simple and robust in dense data scenarios, they incur substantial computational costs in real-time due to continuous recalculations as new data is integrated. Model-based CF, primarily via matrix factorization, mitigates scaling challenges by representing interactions inside latent factor spaces. A further effective method in recommendation systems, notably enhanced by Koren, Bell, and Volinsky, involved decomposing of the interaction matrix into lower-dimensional representations [12].

Moreover, researchers presented Bayesian Personalized Ranking (BPR) [13], which modifies matrix factorization to prioritize ranking over rating prediction, significantly improving the efficacy of CF in implicit feedback contexts. Additional advancements, such as Alternating Least Squares (ALS) [14], have advanced model-based CF by employing efficient parallelizable optimization methods to boost scalability. To overcome these restrictions, current research has investigated alternative models that integrate the advantages of both memory-based and model-based techniques, with the objective of improving recommendation accuracy and scalability while reducing the effects of data sparsity and cold start issues.

### *2.2 Graph-Based Collaborative Filtering*

In CF, another effective method was applied to Graph RSs. Graph-based methodologies surpass conventional techniques by representing user-item interactions as a bipartite graph, augmenting collaborative filtering by capturing higher-order connections [15]. Graph Laplacian Regularization techniques improve recommendation efficacy by smoothing node properties throughout the graph; nevertheless, they frequently neglect subtle intra-graph patterns [16]. The advent of GCNs has transformed this paradigm by consolidating neighborhood information to acquire advanced node representations. Neural Graph Collaborative Filtering (NGCF) employs higher-order connections to disseminate user-item interactions throughout the graph, enhancing node representations [17]. Recent developments in graph methodologies, such as LightGCN, compress the graph convolution procedure by removing feature transformations and non-linear activations, emphasizing efficient neighborhood aggregation [18]. This simplification reduces computing requirements while improving performance.

Additionally, GraphSAGE introduced inductive node embedding by sampling and aggregating local neighborhood features, proving invaluable in large-scale systems where new nodes frequently emerge [19]. Moreover, due to the ability to handle more complex user-item interactions, Huang et al. [20] designed a model that extended into a dual light graph convolution network for discriminative recommendation (Dual-LightGCN). The author proposed a model that filtered out items disliked by users to ensure more discriminative recommendations. The model has been divided into two bipartite subgraphs of the original user-item interaction graph. It applied a movie lens dataset with Precision, Recall, and F1 Score metrics to verify the proposed model.

### *2.3 Federated Learning in Recommendation Systems*

FL arose in response to privacy concerns, wherein models are trained locally on user devices, and only updates (model parameters) are shared centrally, guaranteeing that user data remains inside the local environment [21]. The utilization of FL in RSs is comparatively early. Initial studies suggested a federated CF approach that retains user embedding locally to ensure privacy [22]. This strategy encounters problems, including communication costs from frequent parameter exchanges between devices and central servers and significant data heterogeneity that prevents model convergence and generalization across varied user groups [23].

FedRec has further progressed the implementation of FL in RSs by proving that federated models may attain accuracy equivalent to centralized models [24]. They highlighted novel issues intrinsic to federated methodologies: (1) Communication Expenses: Regular updates between the server and clients elevate network congestion, especially in extensive systems with millions of users [25]. Additionally, (2) Model Convergence: Local models may diverge considerably, requiring effective aggregation procedures to guarantee optimal performance of the global model [26]. (3) Restricted Data Access: Decentralized algorithms are unable to completely utilize global data, which may diminish suggestion quality, particularly for cold start users or infrequent items [27]. Recent research has concentrated on surmounting these obstacles. FedPer [28] utilized a personalization-based approach that distributes a common global model among users while customizing a subset of parameters for individual, hence improving adaptability to local data [29]. Similarly, Hierarchical FL [30] presented a multi-tier federated framework that reduces communication expenses by consolidating updates at local cluster levels prior to transmission to the global server, hence enhancing scalability for larger networks.

Additionally, initiatives such as Secure Federated Matrix Factorization [31] investigated privacy-preserving methods using homomorphic encryption to ensure that model updates remain encrypted, thus boosting the security of user data during the training phase. Incorporating advanced graph neural network architecture becomes essential as we delve deeper into enhancements for federated recommendation systems. This leads us to explore how GCNs, skilled at capturing complex relational data, can be integrated with the privacy advantages of FL [32].

### *2.4 Integration of GCNs with Federated Learning*

The integration of GCNs with FL for recommendation systems represents an underexplored yet very promising area [33,34]. GCNs can capture complex relationships and dependencies between users and items within a graph structure, significantly benefiting CF tasks. However, merging GCNs with FL presents new challenges due to the complex nature of graph operations and the decentralized approach of FL [35].

FedGNN pioneers the application of FL principles to graph neural networks for social network recommendations. Using graph structure and node embeddings within a federated setting, FedGNN makes recommendations while ensuring user data privacy [36]. However, this strategy is specifically designed for social networks where interactions primarily occur between users and may not extend effectively to broader collaborative filtering scenarios such as user-item interactions in e-commerce or media services. Combining GCNs with FL presents a fundamental challenge regarding communication overhead for resource-intensive high-dimensional graph embeddings. Thus, FedGraphNN tackles this by improving the communication process with a compression technique and lowering the size of the shared updates. Nevertheless, compression might result in information loss, affecting the model's capacity to accurately represent complex relationships in graphs [37].

Using GCNs in FL-based recommendation systems presents various difficulties: Graph partitioning limits the GCNs' capacity to understand worldwide interactions by allowing users in a federated environment to access only a local view of the graph. Though methods like graph sampling and local graph aggregation have been proposed to solve this restriction, their investigation in FL environments is still lacking. Graph learning protecting privacy: Using graph-based approaches in FL raises further privacy concerns as the graph structure may expose private data. Differential privacy and safe multi-party computation are developed to protect user privacy during training [38]. Their inclusion into GCNs, however, dramatically raises computing requirements and may affect model accuracy.

# 3 Methodology

## *3.1 Problem Statement*

*Collaborative Filtering (User-Item Interaction Graph)*

We give a scenario of a movie recommendation system where $G = (V, E)$ is a graph representing user-item (movie) interactions. The set of nodes $V$ is partitioned into two disjoint subsets: users $U$ and items $I$, such that $V = U \cup I$. The edges $E \subseteq U \times I$ represent observed interaction between users and items within the system. We use the term "items" to represent the movies in the database collection. Moreover, these interactions can be clicks, watches, views, and duration, which are kinds of implicit feedback (interactions that are not explicit ratings but inferred preferences). Users $(U)$ and Items $(I)$ refer to nodes with an individual user or movie available in the recommendation systems.

CF is an effective method in recommendation systems; however, it encounters challenges when a first-time user joins the system without a previous interaction history. Thus, the system does not deliver items that align with the preference of the target user, and this problem is called a Cold-Start Problem [39]. There are two prevalent types of cold-start scenarios: (1) User Cold-Start happens when a subset. $U_{cold} \subset U$ of users who are new to the system and have no previous interactions. As a result, the system struggles to provide movie recommendations that match their tastes. (2) Item Cold-Start occurs similarly with new movies, indicated as a subset $I_{cold} \subset I$ of items (movies) with no interactions, which impedes the system's ability to recommend these movies effectively.

The cold-start problem undermines the effectiveness of traditional collaborative filtering recommendation systems, which rely heavily on historical interaction data for accurate predictions. The goal is to predict unobserved user-item interactions, including those involving cold-start users and items (movies), by developing significant representations through a GCN model. Additionally, knowledge is consolidated in a federated framework across decentralized datasets maintained by various clients, thus preserving data privacy without requiring centralized data storage.

## *3.2 Proposed Model Architecture*

This section introduces the architecture of our proposed model. The entire mechanism illustrated in Fig. 1 is briefly described in Sections 3.2.1 to 3.2.5.

### *3.2.1 Data Loading and Preprocessing*

We begin with the MovieLens 1M dataset [40], which comprises approximately one million ratings from 6040 users on 3900 movies. This dataset offers a rich source of user-item interactions essential for collaborative filtering in our recommendation system. (1) To efficiently construct the graph and facilitate indexing in our GCN model, we remap the original user and movie IDs to a continuous range starting from zero, optimizing processing. This remapping aids in referencing nodes in matrices and tensors during model training and inference. (a) User IDs: We assign new IDs to users, ranging from 0 to the number of users minus one. For example, with 6040 users, their new IDs range from 0 to 6039. Then (b) Item IDs: We assign new IDs to movies ranging from 0 to the total number of movies minus one. With 3900 movies, their new IDs range from 0 to 3899. Next, (2) User Feature Processing: To enrich user representations and increase the model's understanding of user preferences, we process demographic features: (a) Gender: We encode gender using label encoding. For example, 'Male' is encoded as 0 and 'Female' as 1. (b) Age: We utilize one-hot encoding for different age groups. Each age group is represented by a unique position in a binary vector, where the user's age group is marked as 1, and all others as 0. (3) Occupation: Occupations are encoded using one-hot encoding, where each profession is given a unique position in a binary vector. This allows the model to differentiate users based on their professional backgrounds. For movies, we extract features based on their genres to characterize each item. Genres: Movies often belong to multiple genres. We separate the genre information (e.g., 'Action|Adventure') and apply one-hot encoding to represent the presence of each genre in a movie. Each genre is assigned a position in a binary vector, with a 1 indicating the genre's presence in the movie. This feature extraction helps the model understand the content attributes of each movie.

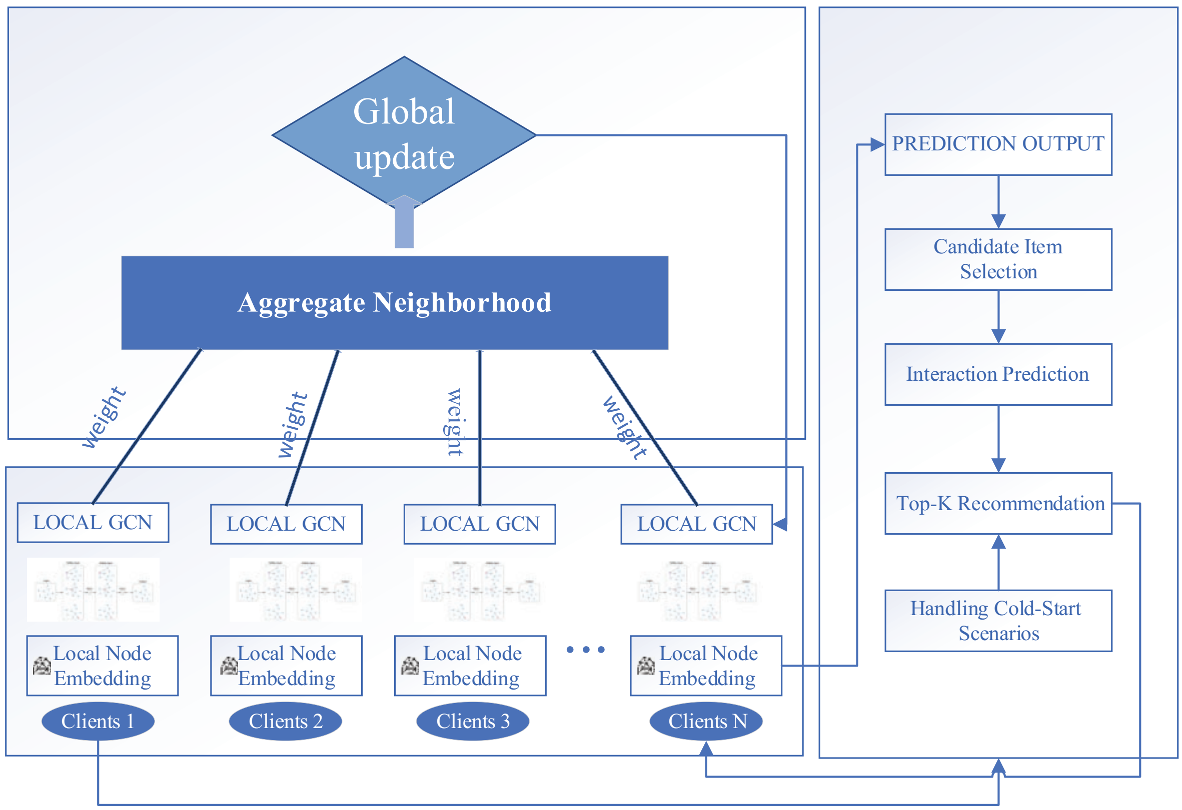


**Figure 1:** The architecture of the proposed model

### *3.2.2 Graph Construction with User and Item Features*

A graph is constructed to represent the interactions between users and movies, incorporating the processed features. For each interaction, like watching a movie, two bidirectional connections are established: (1) From User to Item: An edge is drawn from the user node to the movie node. (2) From Item to User: Conversely, an edge is drawn from the movie node back to the user node. Unique indices are assigned to item nodes to maintain distinctiveness, with offsets applied to item indices.

Subsequently, a unified feature matrix combining user and item features is created: (1) The user feature matrix and the item feature matrix are merged into a single matrix, representing all nodes in the graph, where each row corresponds to either a user or an item. (2) As users and items may possess different feature sets (e.g., users may have features like age and gender, while items may have genres), consistency is ensured by adjusting feature counts. In the case of an imbalance in feature amounts, 0 will add as padding to the set with fewer features, ensuring uniform vector lengths are essential for model processing.

### *3.2.3 GCNs for Implicit Feedback CF*

GCNs utilize an interactive feature propagation scheme to generate accurate embeddings for users and items $Z$ in the collaborative filtering task. GCNs usually stack multiple convolution layers to capture high-order features, effectively enhancing their representation ability. Furthermore, the GCNs model is designed to handle implicit feedback tasks. We use two GCN layers. The propagation rule for each layer $l$ is defined as follows:

$$H^{(1)} = ReLU(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(0)}W^{(0)}) \tag{1}$$

where $\hat{A} = A + I$ refers to the adjacency matrix with self-loop added ($I$ is the identity matrix) and $\hat{D}$ is the diagonal degree matrix of $\hat{A}$ and $W^{(l-1)}$ is the learnable weight matrix of layer $l - 1$. $H^{(l)}$ is the matrix of node embedding at layer $l$. In the first GCN layer ($l = 1$), we input node features. $H^{(0)}$ by computing

$$H^{(1)} = ReLU(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l-1)}H^{(0)}W^{(0)}) \tag{2}$$

which means that the first layer aggregates features from immediate neighbors (including self-loops) and transforms them into a new feature space using the weight matrix $W^{(0)}$. The ReLU (Rectified Linear Unit) activation introduces non-linearity, enabling the model to capture complex patterns. In the second GCN layer ($l = 2$), node embeddings from the first layer $H^{(1)}$ are computed by

$$H^{(2)} = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(1)}W^{(1)} \tag{3}$$

This second layer further propagates and transforms the features, capturing higher-order neighborhood information. No activation function is used here, maintaining the embeddings in a continuous space for subsequent tasks. The final node embeddings $Z = H^{(2)}$ represent the learned representations for users and items.

We aim to predict the likelihood of user-item interactions without explicit ratings to handle Implicit Feedback. Using Negative Sampling to create a balanced dataset, our model addresses this by pairing users with items they have not interacted with (negative samples). For each positive interaction (user $u$ has interacted with item $i$), we randomly sample a set number of items that $u$ has not interacted with, treating these as negative samples. This approach is crucial for training when the model learns to distinguish between preferred and non-preferred items. Next, in Link Prediction, for each user-item pair $(u, i)$, we extract embeddings from $z_u$ and $z_i$. We then concatenate these embeddings to form $h_{ui} = [z_u \| z_i]$. And compute $\hat{y}_{ui}$ Using the link prediction module.

For model training, we use the weighted binary cross-entropy loss function to determine the loss for all samples by

$$\mathcal{L} = -(y_{ui}\log(\hat{y}_{ui}) + (1 - y_{ui})\log(1 - \hat{y}_{ui})) \tag{4}$$

where $y_{ui} = 1\ if\ (u, i)$ indicates a positive interaction; $y_{ui} = 0$ for negative samples. Additionally, we apply to instances where we need to balance weight loss between positive and negative samples.

$$\mathcal{L} = -(w^{+}y_{ui}\log(\hat{y}_{ui}) + w^{-}(1 - y_{ui})\log(1 - \hat{y}_{ui})) \tag{5}$$

which represents $w^{+}$and $w^{-}$, referring to weights assigned to positive and negative samples, respectively. We set $w^{+}$Higher to prioritize the accurate prediction of positive interactions.

*3.2.4 Setup of Federated Learning Framework*

In recognition of the importance of user privacy, we adopt a FL framework to ensure privacy and simulate a decentralized data environment. We distribute the dataset (Movielens) among multiple simulated clients: (1) Users are divided into $K$ disjoint subsets. Each client $k$ holds data corresponding to users in $U$, including their interaction and demographic features. The setup mimics real-world scenarios where data is distributed across different devices. Each client performs local training (Local Training on Client) on its subset of data: (1) Initialization: It is when clients initialize their local models with global model parameters. $\theta^t$ at round $t$. Then, (2) clients train their models using the same GCN architecture and loss function during the local updates model. The process of local training involves multiple epochs over the client's data. (3) Model Updates: After local training, clients obtain updated model parameter $\theta_k^{(t+1)}$. Next, for model aggregation, at the server, the global model is updated by aggregating the client's local models as

$$\theta^{(t+1)} = \sum_{k=1}^{K} \frac{n_k}{n_{total}} \theta_k^{(t+1)} \tag{6}$$

where $n_k$ is the number of samples at client $k$ and $n_{total}$ is the total number of samples across all clients. This weighted averaging ensures that clients with more data have a proportional influence on the global model. Which is then updated global model parameter (Global model update) $\theta^{(t+1)}$ are then distributed back to clients for the next round of training. This process continues interactively for the predefined number of rounds until it converges.

*3.2.5 Recommendation Generation*

With global model trains, we generate personalized recommendations for each user. Fig. 2 illustrates the recommendation generation flow. These processes start with learned embeddings from the GCNs model to predict which items (movies) a user is likely to interact with, even if they have not previously interacted with them. (1) Embedding Computation: The objective is to obtain vector representations (embeddings) for all users and items that capture their characteristics and relationships learned during training. We use the trained GCNs model to compute embeddings for each user, reflecting their preferences based on past interactions and demographic features. Similarly, we compute embeddings for each movie (item), incorporating information about their attributes, like genres. (2) Candidate Item Selection: To identify movies a user has not seen or interacted with, we ensure that recommendations are included with new and relevant content information. We construct a list of candidate movies for each user by excluding those they have already interacted with during the training phase. This list represents potential recommendations that the user might find interesting. Then, (3) Interaction Predictions: We estimate the likelihood that a user will interact with each candidate movie. For this, each user and candidate movie pair combine their embeddings to create a feature representing the potential interaction. We input this combined feature into a prediction function that outputs a score indicating how likely the user is to enjoy or engage with the movie. The higher score determines the high probability of the user liking the movie. Next, (4) We conduct a recommended list of K (top K recommendation) to select the most suitable movies to recommend to each user based on the predicted interaction score probabilities. The process is to rank all candidate movies for each user in descending order of their predicted scores; we then select the top K movies for this ranked list to recommend to the user. Finally, to handle the cold-start scenarios, we integrate the user's demographic features (age, gender, occupation) to generate their embedding. By quickly comparing these features with those of existing users, the model can infer preferences and recommend movies accordingly. Moreover, we use movie attributes (genres) to create their embeddings, allowing the model to predict which users might be interested in these new movies based on their preferences for similar attributes.

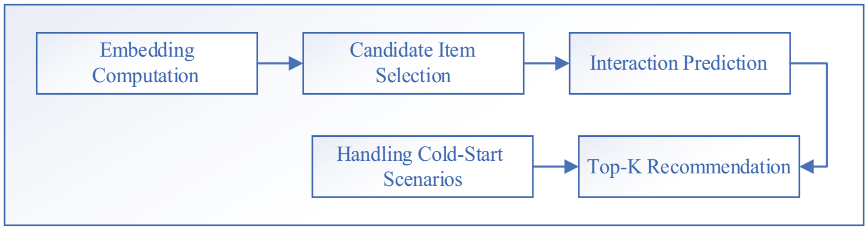


**Figure 2:** Recommendation generation flow

## 4 Experiments

### *4.1 Dataset and Implementation Setup*

We use a real-life dataset called Movielens, which comprises 1 million ratings from 6000 users on about 4000 movies, including the user demographic information and movie genres. We have divided the Movielens dataset into 80% for training and 20% for testing to conduct a recommendation test on cold-start users and items. Additionally, in Federated scenarios, the client's data is distributed across n simulations clients; in our study, we choose 10 simulation rounds. Adapting to accurate simulation, we tune hyperparameters such as embedding dimensions, learning rates, dropout rates, and weight decay, saving the configurations that yield the best performance.

### *4.2 Performance Metrics*

In this study, we select six measures to validate the ranking performance of our recommendation model, each chosen for its ability to reflect different aspects of model performance in the context of a movie recommendation system using the Movielens dataset. The setup metrics include Precision, Recall, F1 Score, NDCG, MRR, and ROC-AUC score.

- Precision evaluates the model's accuracy in making positive predictions, determining how many recommended items are relevant to the users.
- Recall measures the fraction of relevant items the model successfully retrieves out of all available relevant items. The result of high recall is essential in ensuring that users are exposed to a comprehensive list of items that might interest them, thereby enhancing user engagement.
- F1 Score provides a balance between precision and recall, which provides a single metric that conveys the overall accuracy of the recommendation system when both the false positives and false negatives carry a significant but equivalent weight between precision and recall.
- NDCG focuses on the quality of rankings, especially at higher ranks, assessing the model's ability to identify relevant items and rank them in the order of their relevance. This metric decision is particularly pertinent in environments where the top-listed recommendations may significantly impact user choice.
- MRR is a specialist in the model's effectiveness in ranking items, focusing on the rank of the first relevant recommendation. The measure is significant in use cases where the first impression is crucial, such as in online streaming services where the first few recommendations must effectively capture the user's interest.
- ROC-AUC assesses the model's capability to discriminate between classes effectively. High ROC-AUC values indicate that the model is excellent at distinguishing between relevant and non-relevant items, a crucial capability in ensuring user trust and satisfaction.

For baseline models, we compare four metrics, including NDCG, Recall, Precision, and MRR, due to the limitations of the provided baseline libraries. The choice of these metrics allows us to focus on the practical aspects of recommendation systems—ensuring users receive both accurate and relevant content recommendations.

### *4.3 Results of the Proposed Methodology*

Table 1 presents our model's recommendation results at K = 5, K = 10, and K = 20. There is a performance difference between different evaluation metrics across several measures in our recommendation model.

**Table 1:** Performance results of recommendation

| Our model | ROC-AUC | NDCG | Precision | Recall | F1 Score | MRR |
|---|---|---|---|---|---|---|
| K = 5 | 0.4914 | 0.2186 | 0.282 | 0.2058 | 0.237 | 0.3636 |
| K = 10 | 0.7303 | 0.1975 | 0.479 | 0.4792 | 0.479 | 0.3817 |
| K = 20 | 0.8140 | 0.1802 | 0.470 | 0.8760 | 0.612 | 0.3890 |

First, the ROC-AUC values increase as the recommended items increase from 0.4914 at K = 5 to 0.7303 at K = 10 and finally to 0.8140 at K = 20. This trend suggests that the model better distinguishes relevant items as more items are recommended.

Second, NDCG, which evaluates the ranking quality (i.e., how well the top-ranked recommendations match the most relevant items), decreases from 0.2186 at K = 5 to 0.1802 at K = 20. This drop indicates that the model struggles to keep the most relevant items at the top of the list when the list length increases.

Precision, measuring the fraction of the relevant recommended items, rises from 0.282 at K = 5 to 0.479 at K = 10, then dips slightly to 0.470 at K = 20. Meanwhile, recall, which measures what proportion of all truly relevant items are successfully recommended, improved significantly, especially at K = 20, reaching 0.8760. The result suggests that as we recommend more items at K = 20, the model captures a larger share of the relevant items overall, though its precision decreases slightly because we also include a small fraction of non-relevant items in those recommendations.

The F1 Score, which combines the balance of precision and recall, peaks at 0.612 at K = 20. Mean Reciprocal Rank (MRR), which measures how highly the first relevant item is ranked, also improves across the different K values, reaching 0.3890 at K = 20. These improvements in such metrics imply that the model is increasingly able to recognize and correctly identify relevant items as more recommendations are provided.

However, declining NDCG scores across higher K reveal a significant area for further Research, as the system could not satisfy all the evaluation metrics provided and the model's ranking process. Thus, enhancing the model's ability to place the most relevant items at the highest point of the recommended list is critical for increasing its utility in real-world situations. Fig. 3 visually represents the performance metrics described in Table 1.

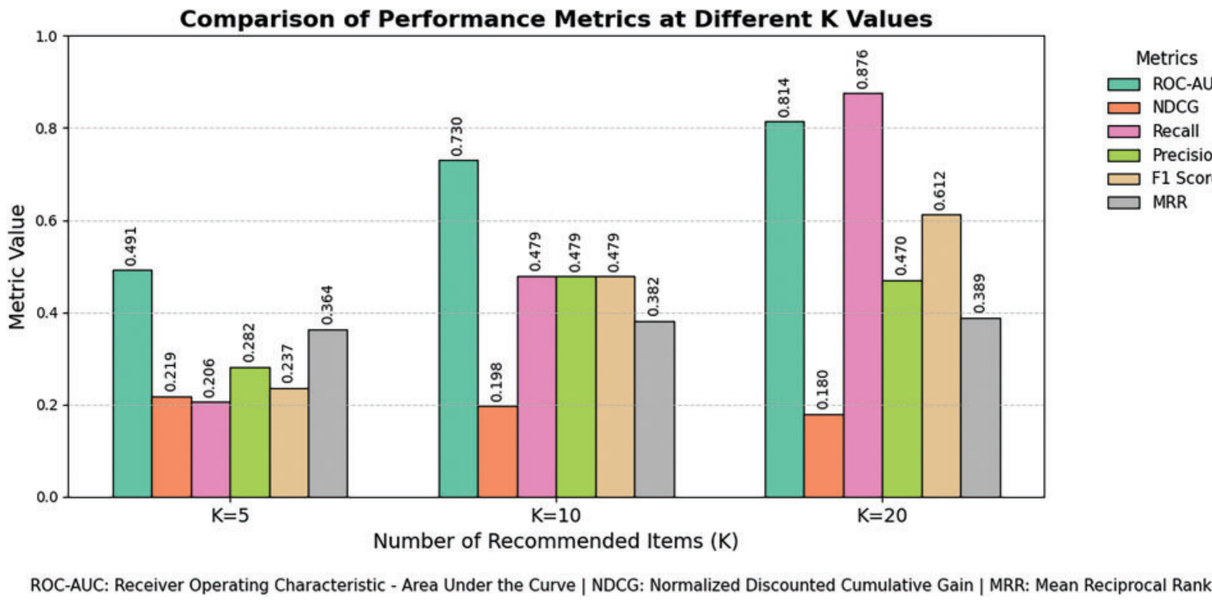


**Figure 3:** Performance metrics of recommendation systems

Fig. 4 illustrates the training loss metrics in a FL context with 10 clients. The graph on the left depicts the training loss across 200 epochs for each client, demonstrating a steady and swift reduction in losses for all users, signifying successful model convergence. The graph illustrates the mean training loss for all clients across 20 federated rounds, demonstrating a significant decrease in loss that stabilizes after approximately 7 rounds. The results indicate a strong FL system in which all clients attain comparable enhancements in model correctness, illustrating the system's effectiveness in minimizing overall error rates within a decentralized training framework.

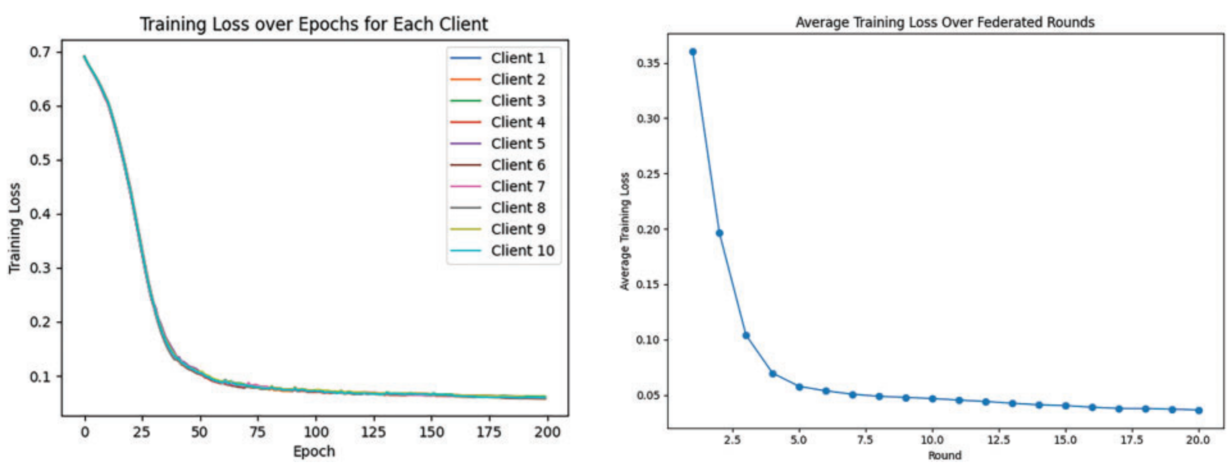


**Figure 4:** Training loss over epochs for each client and average training loss of clients

*Discussion on Precision Trend*

In many conventional RSs, precision typically decreases dramatically as K grows because adding more items often means including more irrelevant ones. To effectively handle implicit feedback, our model employs a negative sampling strategy. For each positive interaction (where a user engages with an item), we randomly sample a set number of items the user has not interacted with, treating them as negative samples. This approach balances the dataset and helps the model distinguish between preferred and non-preferred items better. Moreover, our study prioritizes these positive interactions as the basis for generating recommendations. The embeddings acquired from these interactions faithfully reflect user preferences, therefore guaranteeing that objects like those already used get better ranking scores. Please refer to Section 3.2.5 for the specifics on the recommendation generating procedure.

### *4.4 Baseline Detail and Performance Results*

In this section, we compare our method against the following baselines: (1) NGCF [17], which enhances traditional matrix factorization by integrating neural networks to model complex, non-linear user-item interactions; (2) LightGCN [18]; (3) DeepFM, a hybrid recommendation model combining factorization machines (FM) and deep neural networks (DNN) [41]; and (4) BPR [13]. Table 2 presents a baseline performance comparison. We used the default hyperparameters originally set by the authors for a fair comparison. Moreover, most of the models were sourced from libraries, and we did not change any parameters; all remained the same as initially set by the authors.

**Table 2:** Performance result of baselines

| Models | NDCG | Recall | Precision | MRR |
|---|---|---|---|---|
| NGCF | 0.1855 | 0.1597 | 0.1297 | 0.3547 |
| LightGCN | 0.1589 | 0.1342 | 0.1112 | 0.3154 |
| DeepFM | 0.1809 | 0.1552 | 0.1292 | 0.3433 |
| BPR | 0.1933 | 0.1671 | 0.1352 | 0.3633 |
| Proposed model | 0.1802 | 0.8760 | 0.4700 | 0.3890 |

Among the recommended K items, our system achieves its best result at K = 20, excelling in recall, precision, and MRR. Additionally, it outperforms LightGCN in NDCG but falls short of the other three models (BPR, NGCF, DeepFM). While most models show robust performance in MRR, precision and recall are generally low, except for our proposed model, which attains a high recall of approximately 0.876, indicating a strong match between the recommended relevant ranks (positives) and user preferences. Furthermore, our model performs well in recall and precision compared to the other four baseline models, as illustrated in Table 2 and the performance visualization in Fig. 5.

The integration of GCN within the CF framework in our proposed method effectively captures more complex, non-linear user and item relationships. This, in turn, allows the system to predict user preferences more accurately, increasing the likelihood that recommended items are both relevant (improving precision) and cover a more significant portion of relevant items (improving recall). While existing baselines (e.g., NGCF, LightGCN) also leverage graph structures or neural components, our model refines how user–item information is incorporated and propagated—especially regarding integrating other advanced FL models—which enhances overall robustness, security, and performance.

Since we intentionally kept all default parameters for the baseline models, these models may remain unadapted to the dataset's complexity or lack sufficient model capacity. In contrast, our model has been tuned to emphasize metrics such as recall and precision. By carefully aligning model hyperparameters and loss functions to capture relevant items more accurately, we achieve higher performance on these metrics. Additionally, our proposed method is trained to distinguish between relevant and irrelevant items more effectively than the baseline methods through effective handling between positive and negative sampling strategies and user preference distribution. Consequently, the model is more likely to retrieve items of true interest and exclude those of lower relevance, boosting recall (by returning more positive hits) and precision (by ensuring higher relevance among returned items).

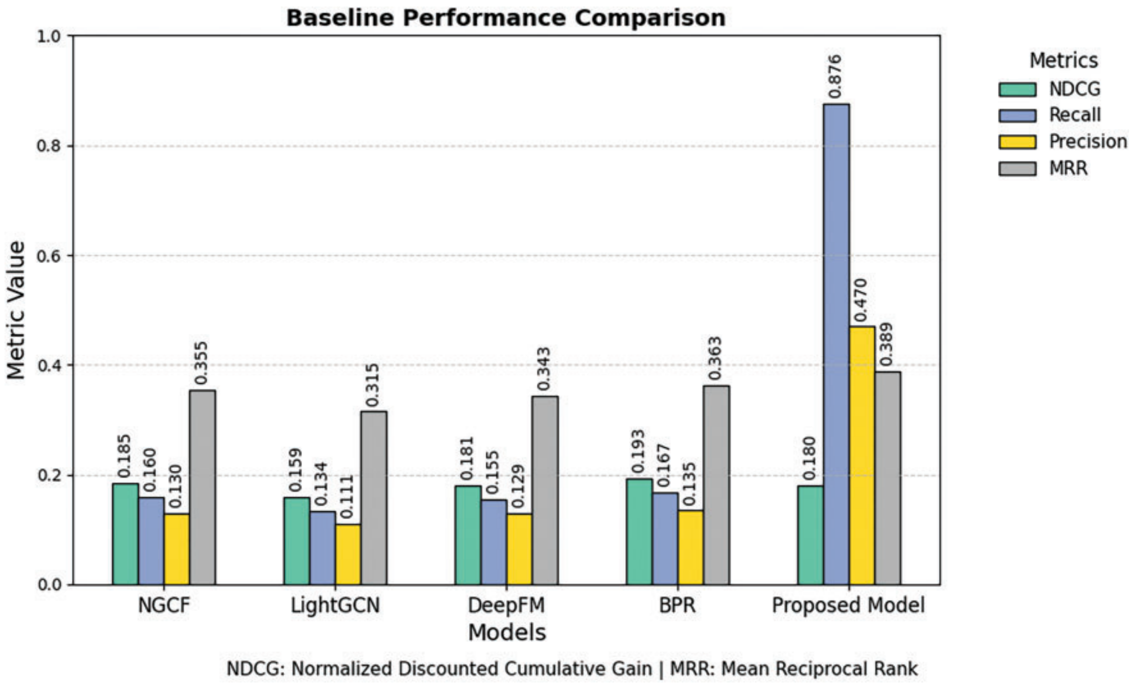


**Figure 5:** Proposed model and baseline model comparison

## 5 Ablation Study of Proposed Model

This section evaluates the effectiveness of our recommendation system compared to centralized training, considering the influence of FL. Another objective is understanding the trade-offs between model accuracy and user privacy. We conduct further experiments using both centralized and federated training methods. Both setups aim to maintain the consistency of the model architecture and hyperparameters to ensure a fair comparison. The new setup consists of the embedding dimension (128), learning rate (0.001), number of rounds (20), dropout (0.5), weight decay ($1e-5$), and negative sampling ratio (1.0) were uniformly set. The model was trained on the full dataset without data partitioning in centralized training. We simulated 10 clients for FL, each holding a subset of the user-item interaction data.

**Result and Analysis:** In centralized training, the model accessed the entire dataset throughout the training process. The model could potentially identify more complex patterns and correlations by leveraging the full spectrum of user-item interactions. In the FL setup, the data was partitioned among 10 clients, and the model was trained collaboratively without necessitating the provision of raw data by clients. Each client trained the model using local data and then transmitted updated model versions to a central server for integration. On all metrics, the federated approach slightly underperforms the centralized model. For instance, the federated model values about 0.3750, whereas the centralized model values around 0.7931. Compared to the centralized approach, the federated paradigm reduces performance slightly. While federated training offers benefits like privacy and localized computations, it might face challenges in matching the predictive performance of centralized approaches, potentially due to data distribution variances or limited model updates from each client. The results indicate a trade-off between maintaining data privacy and achieving optimal model performance. Fig. 6 illustrates the comparison of performance metrics between federated and centralized training.

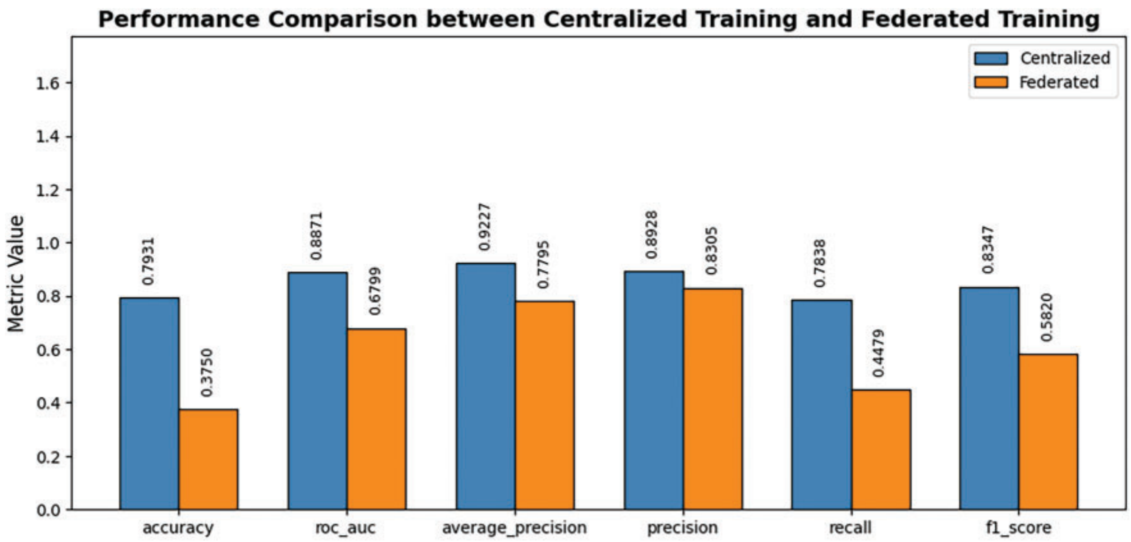


**Figure 6:** Comparison of centralized training and federated learning

## 6 Conclusion and Future Perspectives

Integrating FL and GCNs into CF significantly advances recommendation systems, emphasizing user privacy and enhancing system performance. This approach employs GCNs to dissect complex relational data and FL to maintain data locality, effectively addressing privacy and sovereignty concerns. Furthermore, the system alleviates issues inherent in traditional collaborative filtering by improving prediction accuracy. It supports privacy-centric processing of sensitive data and provides personalized recommendations [42] using diverse data sources, all while ensuring user privacy is protected [43]. Additionally, our approach has demonstrated better performance compared to existing benchmarks.

Despite these developments, our results in the ablation study show that centralized training approaches still outperform federated models in numerous performance criteria, including accuracy, recall, and precision. This scenario draws attention to a possible trade-off whereby a slight decrease in model performance might offset the advantages of more privacy and data protection in FL. Still, especially in privacy-centric processing and user-centric personalizing, the federated approach has clearly shown benefits over conventional techniques.

Future research will improve scalability, apply homomorphic encryption [44], handle non-IID data, fit dynamic graphs, and provide cross-domain recommendations by other advanced privacy-preserving technologies. Also, to make the model of FL and GCNs more valuable and practical, it would be better to work on creating new ways to measure privacy, speed, and personalization with shared context. It is also necessary to include more real-world datasets to improve the results of the recommendations.

**Acknowledgement:** I express my sincere gratitude to all individuals who have contributed to this paper. Their dedication and insights have been invaluable in shaping the outcome of this work.

**Funding Statement:** This research was funded by Soonchunhyang University, Grant Numbers 20241422 and BK21 FOUR (Fostering Outstanding Universities for Research, Grant Number 5199990914048).

**Author Contributions:** The authors confirm contribution to the paper as follows: study conception and system design: Sony Peng and Kyuwon Park; supervisors: Dae-Young Kim and Doo-Soon Park; data preprocessing: Sony Peng, Sophort Siet, and Ilkhomjon Sadriddinov; analysis and interpretation of results: Sony Peng, Dae-Young Kim, Kyuwon Park, and Doo-Soon Park; draft manuscript preparation: Sony Peng, Dae-Young Kim, Kyuwon Park, and Doo-Soon Park. All authors reviewed the results and approved the final version of the manuscript.

**Availability of Data and Materials:** We used publicly available data and gave a reference to it in our paper. Please check reference [40].

**Ethics Approval:** Not applicable.

**Conflicts of Interest:** The authors declare no conflicts of interest to report regarding the present study.

## References

1. Kumar P, Thakur RS. Recommendation system techniques and related issues: a survey. Int J Inf Technol. 2018;10(4):495–501. doi:10.1007/s41870-018-0138-8.
2. Zangerle E, Bauer C. Evaluating recommender systems: survey and framework. ACM Comput Surv. 2023;55(8):1–38. doi:10.1145/3556536.
3. Vilakone P, Park DS, Xinchang K, Hao F. An efficient movie recommendation algorithm based on improved k-clique. Hum Centric Comput Inf Sci. 2018;8(1):1–15. doi:10.1186/s13673-018-0161-6.
4. Himeur Y, Sohail SS, Bensaali F, Amira A, Alazab M. Latest trends of security and privacy in recommender systems: a comprehensive review and future perspectives. Comput Secur. 2022;118(4):102746. doi:10.1016/j.cose.2022.102746.
5. Peng S, Siet S, Ilkhomjon S, Kim DY, Park DS. Integration of deep reinforcement learning with collaborative filtering for movie recommendation systems. Appl Sci. 2024;14(3):1155. doi:10.3390/app14031155.
6. Phonexay V, Xinchang K, Park D. Movie recommendation system based on users' personal information and movies rated using the method of k-clique and normalized discounted cumulative gain. J Inf Process Syst. 2017;16(2):494–507. doi:10.3745/JIPS.04.0169.
7. Sadriddinov I, Park DS, Kim D, Yang Y, Peng S, Siet S. Movie recommendation system using community detection based on the Girvan–Newman Algorithm. In: Advances in computer science and ubiquitous computing. Lecture notes in electrical engineering. Vol. 1028. Singapore: Springer; 2022. p. 599–605. doi:10.1007/978-981-99-1252-0_80.
8. Siet S, Peng S, Ilkhomjon S, Kang M, Park DS. Enhancing sequence movie recommendation system using deep learning and KMeans. Appl Sci. 2024;14(6):2505. doi:10.3390/app14062505.
9. Guo Y, Bo D, Yang C, Lu Z, Zhang Z, Liu J, et al. Data-centric graph learning: a survey. IEEE Trans Big Data. 2025;11(1):1–20. doi:10.1109/TBDATA.2024.3489412.
10. Papadopoulos C, Kollias KF, Fragulis GF. Recent advancements in federated learning: state of the art, fundamentals, principles, IoT applications and future trends. Fut Internet. 2024;16(11):415. doi:10.3390/fi16110415.
11. Peng S, Yang Y, Mao M, Park DS. Centralized machine learning versus federated averaging: a comparison using MNIST dataset. KSII Trans Internet Inf Syst. 2022;16(2):742–56. doi:10.3837/tiis.2022.02.020.
12. Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems. Computer. 2009;42(8):30–7. doi:10.1109/MC.2009.263.
13. Hu Y, Xiong F, Pan S, Xiong X, Wang L, Chen H. Bayesian personalized ranking based on multiple-layer neighborhoods. Inf Sci. 2021;542(4):156–76. doi:10.1016/j.ins.2020.06.067.
14. Chen J, Fang J, Liu W, Tang T, Yang C. clMF: a fine-grained and portable alternating least squares algorithm for parallel matrix factorization. Future Gener Comput Syst. 2020;108(8):1192–205. doi:10.1016/j.future.2018.04.071.
15. Liu M, Li J, Liu K, Wang C, Peng P, Li G, et al. Graph-ICF: item-based collaborative filtering based on graph neural network. Knowl Based Syst. 2022;251(1):109208. doi:10.1016/j.knosys.2022.109208.

16. He L, Wang X, Wang D, Zou H, Yin H, Xu G. Simplifying graph-based collaborative filtering for recommendation. In: Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining; 2023; Singapore: ACM. p. 60–8. doi:10.1145/3539597.3570451.
17. Wang X, He X, Wang M, Feng F, Chua TS. Neural graph collaborative filtering. In: Proceedings of the 42nd International. ACM SIGIR Conference on Research and Development in Information Retrieval; 2019. p. 165–74. doi:10.1145/3331184.3331267.
18. He X, Deng K, Wang X, Li Y, Zhang Y, Wang M. LightGCN: simplifying and powering graph convolution network for recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval; 2020; China: ACM. p. 639–48. doi:10.1145/3397271.3401063.
19. El Alaoui D, Riffi J, Sabri A, Aghoutane B, Yahyaouy A, Tairi H. Deep GraphSAGE-based recommendation system: jumping knowledge connections with ordinal aggregation network. Neural Comput Appl. 2022;34(14):11679–90. doi:10.1007/s00521-022-07059-x.
20. Huang W, Hao F, Shang J, Yu W, Zeng S, Bisogni C, et al. Dual-LightGCN: dual light graph convolutional network for discriminative recommendation. Comput Commun. 2023;204(8):89–100. doi:10.1016/j.comcom.2023.03.018.
21. Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, et al. Advances and open problems in federated learning. arXiv:1912.04977. 2021.
22. Ammad-Ud-Din M, Ivannikova E, Khan SA, Oyomno W, Fu Q, Tan KE, et al. Federated collaborative filtering for privacy-preserving personalized recommendation system. arXiv:1901.09888. 2019.
23. Smith V, Chiang CK, Sanjabi M, Talwalkar AS. Federated multi-task learning. Adv Neural Inf Process Syst. arXiv:1705.10467. 2017.
24. Chen C, Zhang J, Tung AKH, Kankanhalli M, Chen G. Robust federated recommendation system. arXiv:2006.08259. 2020.
25. Bonawitz K, Eichner H, Grieskamp W, Huba D, Ingerman A, Ivanov V, et al. Towards federated learning at scale: system design. Proc Mach Learn Syst. 2019;1:374–88.
26. Zhao Y, Li M, Lai L, Suda N, Civin D, Chandra V. Federated learning with non-IID data. arXiv:1806.00582. 2018.
27. Arivazhagan MG, Aggarwal V, Singh AK, Choudhary S. Federated learning with personalization layers. arXiv:1912.00818. 2019.
28. Yang Q, Liu Y, Chen T, Tong Y. Federated machine learning. ACM Trans Intell Syst Technol. 2019;10(2):1–19. doi:10.1145/3339474.
29. Liu L, Zhang J, Song SH, Letaief KB. Client-edge-cloud hierarchical federated learning. In: ICC 2020—2020 IEEE International Conference on Communications (ICC); 2020 Jun 7–11. Dublin, Ireland: IEEE; 2020. p. 1–6. doi:10.1109/icc40277.2020.9148862.
30. Chai D, Wang L, Chen K, Yang Q. Secure federated matrix factorization. IEEE Intell Syst. 2021;36(5):11–20. doi:10.1109/MIS.2020.3014880.
31. Li H, Cai Z, Wang J, Tang J, Ding W, Lin CT, et al. FedTP: federated learning by transformer personalization. IEEE Trans Neural Netw Learn Syst. 2024;35(10):13426–40. doi:10.1109/TNNLS.2023.3269062.
32. Liu R, Xing P, Deng Z, Li A, Guan C, Yu H. Federated graph neural networks: overview, techniques, and challenges. IEEE Trans Neural Netw Learn Syst. 2024;36(3):4279–95. doi:10.1109/TNNLS.2024.3360429.
33. Yin Y, Li Y, Gao H, Liang T, Pan Q. FGC: GCN-based federated learning approach for trust industrial service recommendation. IEEE Trans Ind Inform. 2023;19(3):3240–50. doi:10.1109/TII.2022.3214308.
34. Campbell A, Liu H, Scaglione A, Wu T. A federated learning approach for graph convolutional neural networks. In: 2024 IEEE 13rd Sensor Array and Multichannel Signal Processing Workshop (SAM); 2024 Jul 8–11. Corvallis, OR, USA: IEEE; 2024. p. 1–5. doi:10.1109/SAM60225.2024.10636596.
35. Wu C, Wu F, Cao Y, Huang Y, Xie X. Fedgnn: federated graph neural network for privacy-preserving recommendation. arXiv:2102.04925. 2021.
36. He C, Balasubramanian K, Ceyani E, Yang C, Xie H, Sun L, et al. A federated learning system and benchmark for graph neural networks. arXiv:2104.07145. 2021.

37. Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, et al. Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security; 2016; Vienna Austria: ACM. p. 308–18. doi:10.1145/2976749.2978318.
38. Li J, Huang H. Fedgrec: federated graph recommender system with lazy update of latent embeddings. arXiv:2210.13686. 2022.
39. Silva N, Carvalho D, Pereira ACM, Mourão F, Rocha L. The pure cold-start problem: a deep study about how to conquer first-time users in recommendations domains. Inf Syst. 2019;80(11):1–12. doi:10.1016/j.is.2018.09.001.
40. Harper FM, Konstan JA. The MovieLens datasets. ACM Trans Interact Intell Syst. 2016;5(4):1–19. doi:10.1145/2827872.
41. Guo H, Tang R, Ye Y, Li Z, He X, Guo H, et al. DeepFM: a factorization-machine based neural network for CTR prediction. arXiv:1703.04247. 2017.
42. Ali W, Din SU, Khan AA, Tumrani S, Wang X, Shao J. Context-aware collaborative filtering framework for rating prediction based on novel similarity estimation. Comput Mater Contin. 2020;63(2):1065–78. doi:10.32604/cmc.2020.010017.
43. Khan M, Glavin FG, Nickles M. Federated learning as a privacy solution—an overview. Procedia Comput Sci. 2023;217:316–25. doi:10.1016/j.procs.2022.12.227.
44. Kumar Singh P, Kanti Dutta Pramanik P, Sardar M, Nayyar A, Masud M, Choudhury P. Generating a new shilling attack for recommendation systems. Comput Mater Contin. 2022;71(2):2827–46. doi:10.32604/cmc.2022.020437.