# On-Device Recommender Systems

## A Tutorial on The New-Generation Recommendation Paradigm

The Web Conference 2024, Singapore

Hongzhi Yin, Tong Chen, Liang Qu
The University of Queensland, Australia

Bin Cui
Peking University, China

THE UNIVERSITY OF QUEENSLAND AUSTRALIA | CREATE CHANGE



*ODRS is a shorthand for On-Device Recommender System.

# The Tutorial Team

Our tutorial is prepared by a team of four:

**Prof. Hongzhi Yin:** Full Professor and ARC Future Fellow at The University of Queensland

**Dr. Tong Chen:** Senior Lecturer and ARC DECRA Fellow at The University of Queensland

**Mr. Liang Qu:** Senior PhD at The University of Queensland, starting Postdoc in Deakin University soon

**Prof. Bin Cui:** IEEE Fellow, Boya Distinguished Professor and Vice Dean in School of CS at Peking University

# Overview of Recommender Systems

**Role of Recommender Systems (RSs):**

- Counteracting information overload
- Providing tailored user experience
- Targeted marketing and advertising

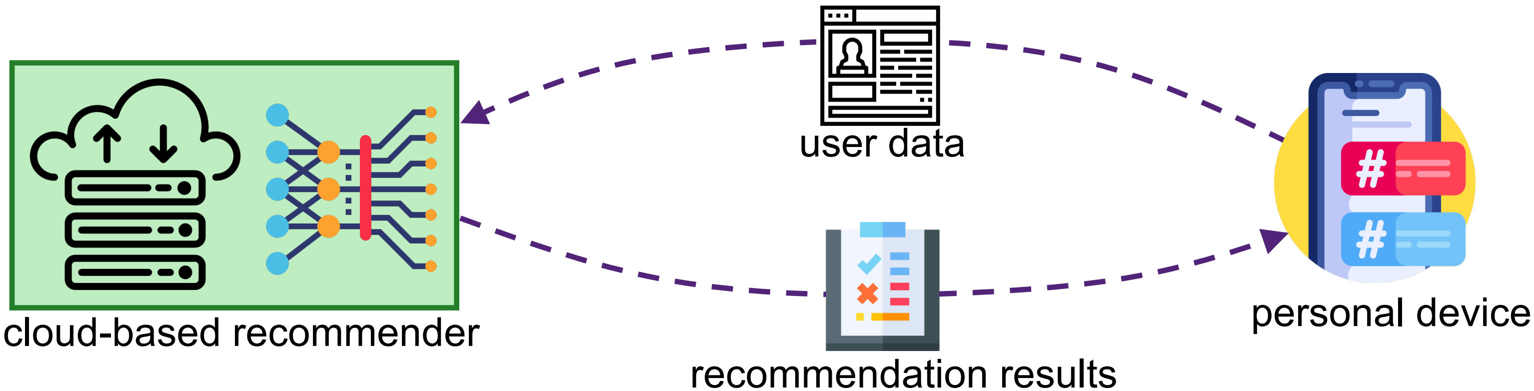
https://flipboard.com

RSs are becoming ubiquitous – let's do an App check on your phone:

And (not so) surprisingly, **they are all equipped with RS algorithms!**

# How Do Current RSs Work?

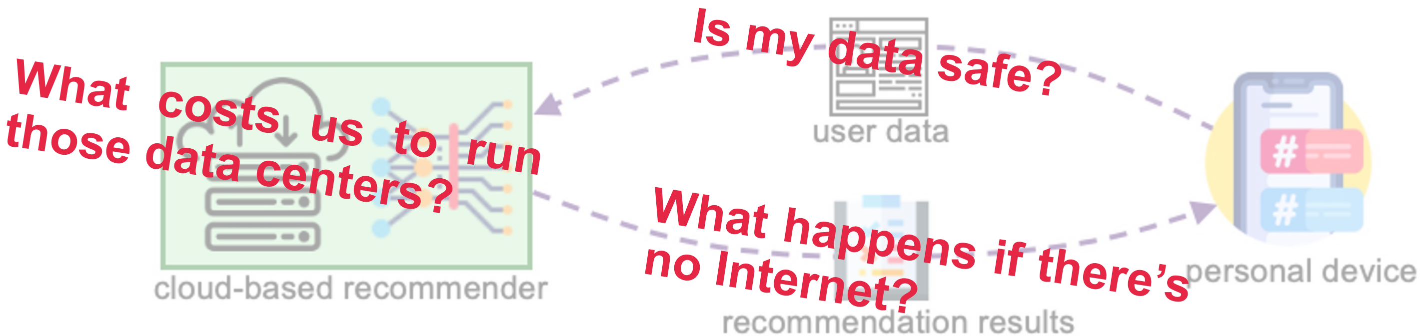
Most existing recommendation architectures are cloud-based.




| Player | Role |
|---|---|
| Cloud-based RS | Computing (training&inference) + interaction data storage |
| Personal device | Data collection + result display |

# Where Things Go Wrong on The Cloud

**Question:** Is this the optimal recommendation paradigm? What can go wrong?




Some stats [OAIC 2013, ACCC 2013, Jones 2018]:

- In Australia, **892** reported data breaches in 2023, with **35% affecting >100 users** and **64%** coming from retail, health, and financing services!
- Optus (a major Aussie mobile service provider) only has **70% 3G geographic coverage** – the number drops to **60%** in remote areas where all the point of interests and great campsites are found.
- Information and communications technology (ICT) is predicted to use **21% of the global electricity**, where data centers accounts for **1/3**.

# Downsides of Cloud-based RSs

With the rise of privacy awareness, need for service timeliness, and promotion for green AI, cloud-based RSs are showing their downsides:

**Privacy** has always been a challenge for RSs [Ge et al 2022].
New privacy legislations like GDPR (EU), CCPA (US), and PIPL (CHN) further bottlenecks the user of customer information [*Noia et al. 2022*].

Current RSs heavily rely on **wireless communication** [Chen et. al. 2021].
Think of the online shopping experience during "Black Friday" sales in the US, or the "double-11" discount campaign – is the timeliness still there?

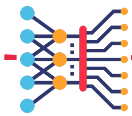
Could-based RSs lead to **huge computation resource cost & energy footprints** [Wang et. al. 2020].Is this affordable for business at all scales? Is this sustainable in a long-term view? Are we wasting the increasing computing power on edge devices?

# A Remedy: On-Device RSs

An emerging research direction: on-device recommender systems, a.k.a. DeviceRSs, or ODRSs [Yin et al. 2024].

**Role of devices:** data collection & display & computing!

deploy locally



lightweight recommender

**Role of the cloud:**

- Job assignment,
- Parameter transporting,
- Version control
- …

Tasks are lighter, no full model training or data handling

(More on it later)

# Promises of ODRSs

As a new recommendation paradigm, ODRSs offer great promises.

**Feature 1:** No need to transmit data elsewhere for analysis → **Private**

**Feature 2:** A recommender is onboard to generate results → **Instant**

**Feature 3:** Most computations are done on small devices → **Resource-efficient**

**Not a fairy tale in academia** – plenty of **R&D outcomes** already:

- Kuaishou – Short Video RS on Mobile Devices [Gong et al. 2022]
- Google – TensorFlow Lite Recommendation [TFL 2024]
- Taobao – The EdgeRec System [Gong et al. 2020]
- Brave Browser – News RS uses Federated Learning [Minto et al. 2021]

快手 KUAISHOU TECHNOLOGY TensorFlow 淘宝 Taobao brave

# ODRSs: What's Important?

In this tutorial, we will cover three technical pillars of ODRSs.

**I. Deployment and Inference:**
"Primitive" solution to ODRSs – Given a complex recommendation model, how do we adapt it to resource-constraint on-device environments?

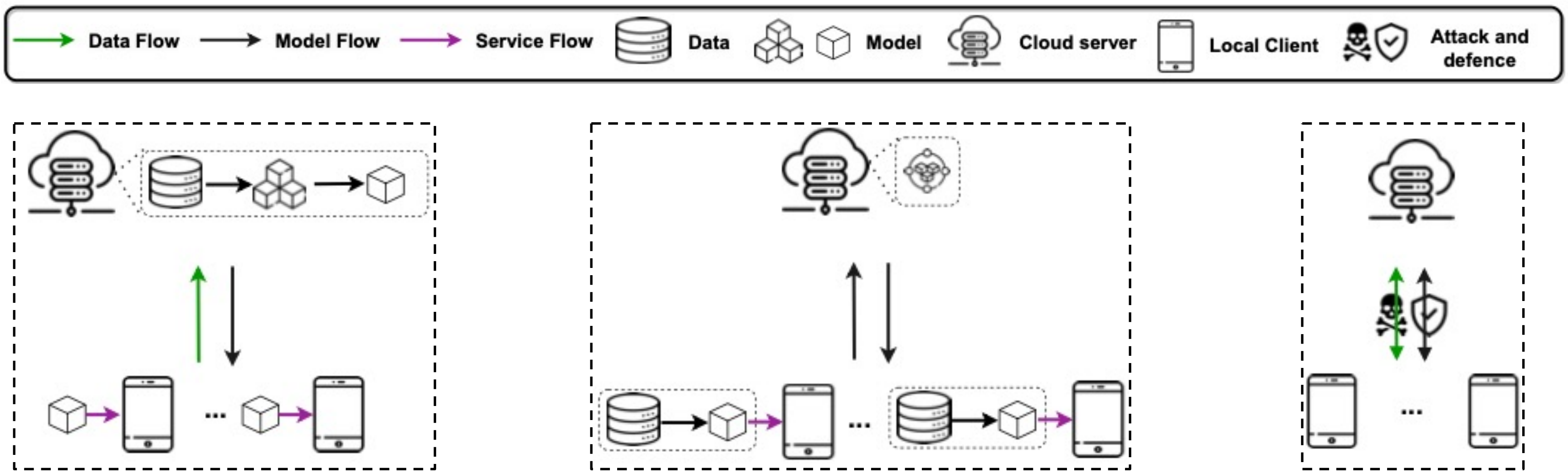
**II. Training and Updating:**
"Ground-up" solution to ODRSs – Can we design models and learning algorithms specific to on-device settings from scratch?

**III. Security and Privacy:**
"Safeguard" for ODRSs – What countermeasures can we take to protect ODRSs from adversaries in the open cyber space?

Now, let's unfold the core of this new recommendation paradigm!

# ODRSs: What's Important? (Cont.)

A visual illustration:




**I. Deployment and Inference**

**II. Training and Updating**

**III. Security and Privacy**

# Definition

**Definition 1 – Traditional, Basic Recommendation:**
**In:** Dataset recording interactions between users $u \in U$ and items $v \in V$.
**Out:** A pairwise similarity function $f(u, v)$, which is trained to capture the affinity between each $(u, v)$ pair.

$f(u, v)$ can be parameterized in multiple ways, such as:

- Matrix Factorization [Koren et al. 2009]
- Factorization Machines [Rendle 2011]
- Neural Collaborative Filtering, e.g., NeuMF [He et al. 2017a]
- Neural Factorization Machines, e.g., NFM [He et al. 2017b]
- Graph Neural Networks, e.g., LightGCN [He et al. 2020]
- And many more…

# Definition (Cont.)

For $f(u,v)$, both the model choice and optimization objective are dependent on the actual task:

our focus today

1. **Top-k recommendation**
   Bayesian Personalized Ranking (BPR) [Rendle et al. 2009] loss:
$$\mathcal{L}_{rec} = \sum\nolimits_{\forall(u,v^+,v^-)} -\log\sigma(f(u,v^+) - f(u,v^-))$$
2. **Click-through rate prediction**
   Negative log-likehood loss [Zhou et al. 2018]:
$$\mathcal{L}_{rec} = \sum\nolimits_{\forall(u,v,y)} -(y\log f(u,v) + (1-y)\log(1-f(u,v)))$$

less so

3. **Rating prediction**
   Squared error loss [Chen et al. 2020]:
$$\mathcal{L}_{rec} = \sum\nolimits_{\forall(u,v,y)} (y - f(u,v))^2$$

# Definition (Cont.)

**Definition 2 (Informal) – Recommendation under On-Device Settings**
Naturally, $f(u,v)$ needs to meet additional on-device requirements.

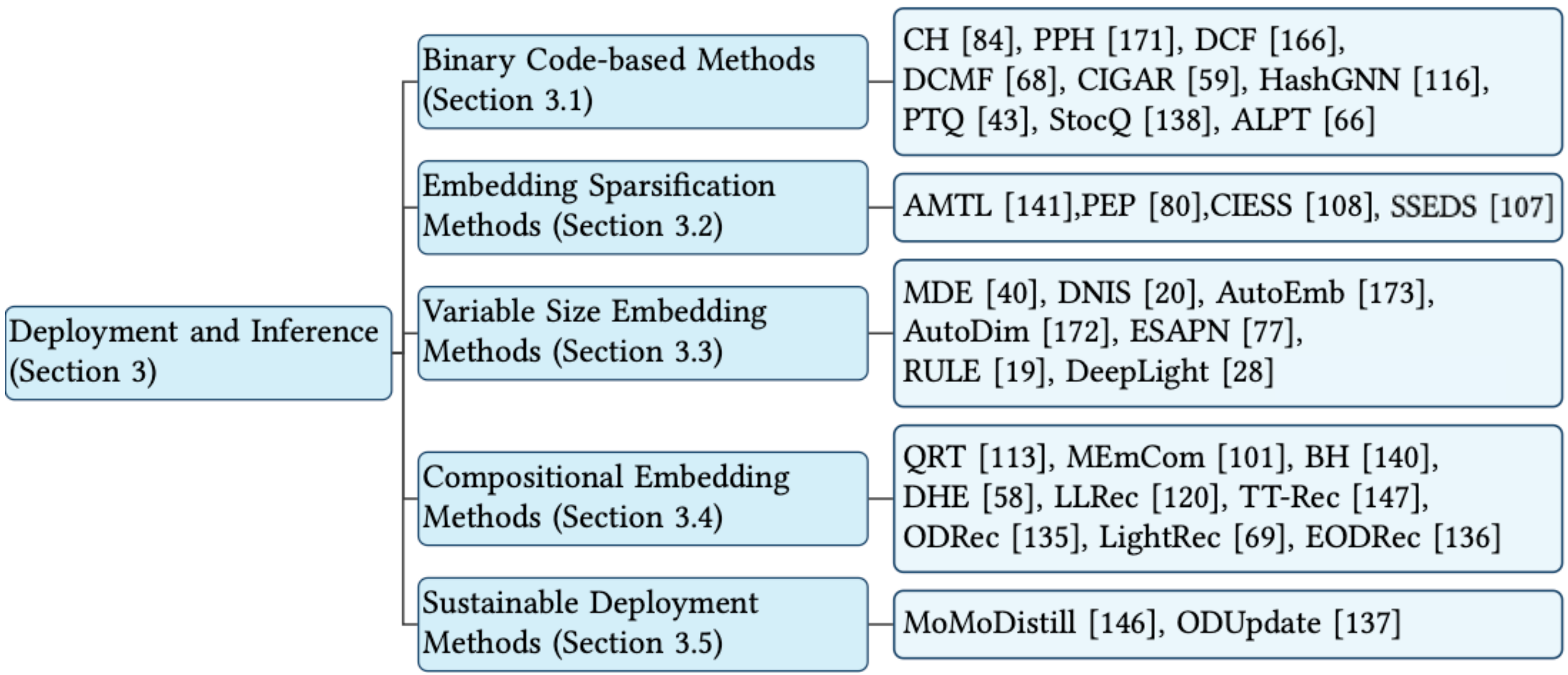
**Deployment and inference – no on-device training needed**
[Parameter size] Can $f(u,v)$ fit in small memory?
[Inference time] Can $f(u,v)$ evaluate quickly on-device?

**Training and updating – real-time on-device training needed**
[Training efficiency] Will training $f(u,v)$ consume much energy?
[Communication overhead] Does $f(u,v)$ frequently exchange info elsewhere?

**Security and privacy – passive and active protection**
[Model privacy] Is the sharable info (e.g., weights) in $f(u,v)$ sensitive?
[Attack resistance] Is $f(u,v)$ robustness to adversarial attacks?

# Taxonomy: Deployment and Inference

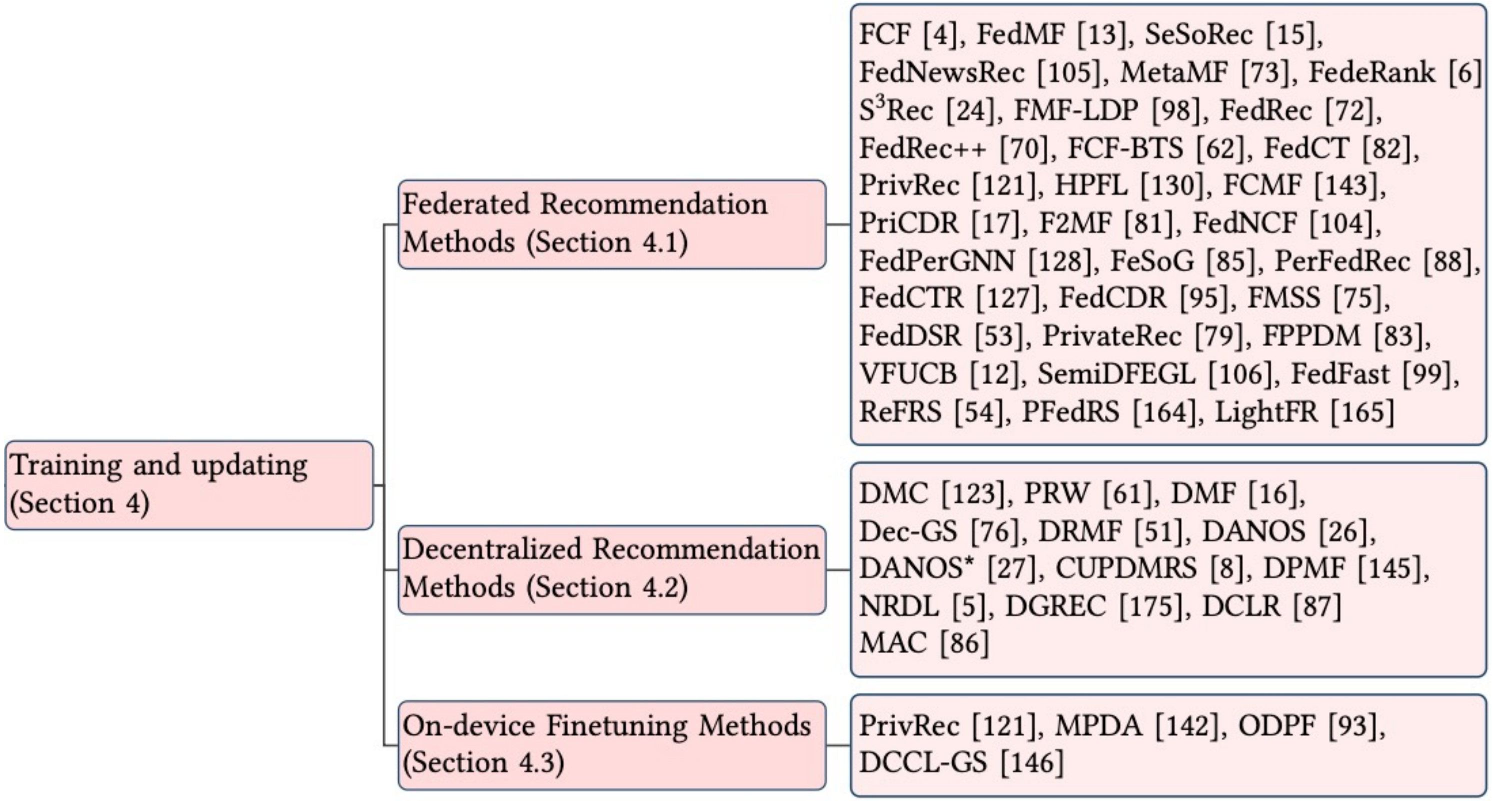
Our taxonomy w.r.t. the deployment and inference of ODRSs.




The corresponding references to different methods can be found in our comprehensive survey [Yin et al. 2024].

# Taxonomy: Training and Updating

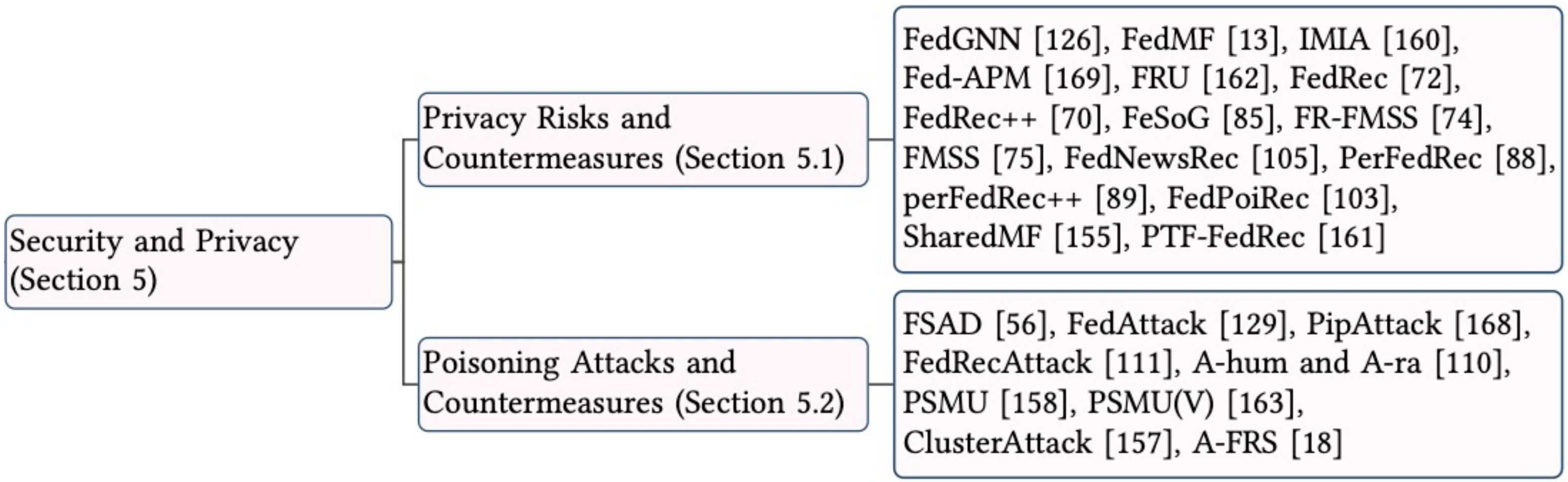
Our taxonomy w.r.t. the training and updating of ODRSs.




The corresponding references to different methods can be found in our comprehensive survey [Yin et al. 2024].

# Taxonomy: Security and Privacy

Our taxonomy w.r.t. the security and privacy of ODRSs.

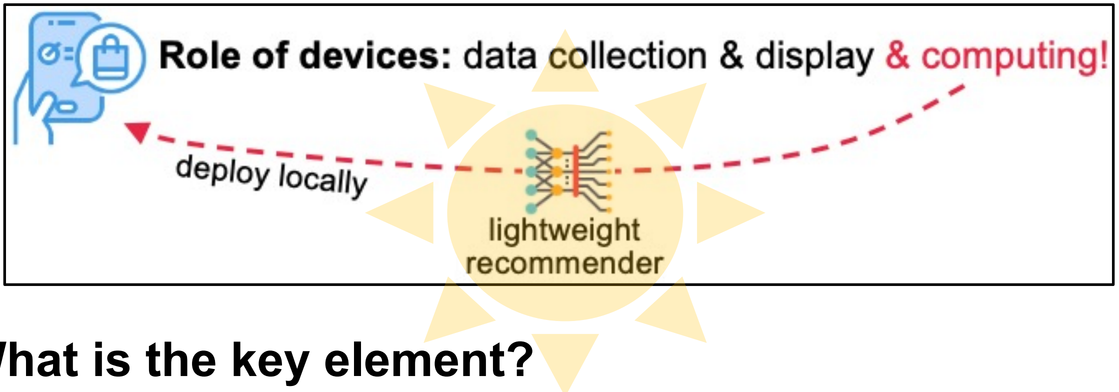



The corresponding references to different methods can be found in our comprehensive survey [Yin et al. 2024].

*ODRS is a shorthand for On-Device Recommender Systems.

# Deployment and Inference: An Overview

Let's have a quick recap on this figure:




**Question: What is the key element?**

The key is to build a lightly parameterized recommender.

Almost all existing studies [Zhang et al. 2016, Joglekar et al. 2020, Liang et al. 2023, Liang et al. 2024] are around embeddings.

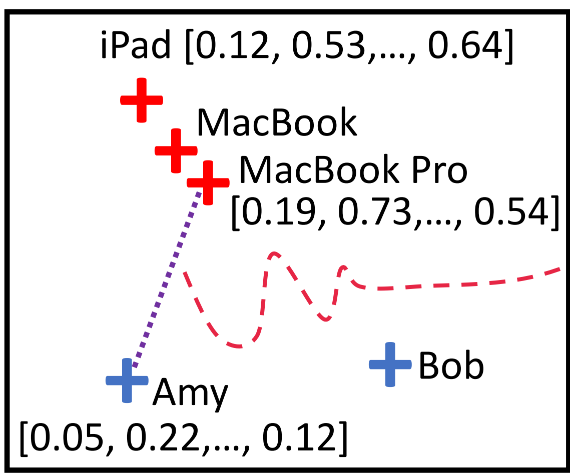
- **What are embeddings?**
- **Why do they matter?**
- **How lightweight embeddings are achieved in ODRSs?**

# Embeddings: What

In RSs, embeddings are $\mathbb{R}^d$ vector representations of **entities**.

Entities in **ID-based recommendation**:
*"IDs" of users and items – just users and items*

Entities in **feature-based recommendation**:
*Features describing users and items, plus their IDs*




User-item affinity can be easily reflected via distance metrics (cosine, dot, Euclidean, etc.)

Embeddings are the main parameter source of recommender $f(u, v)$!

# Embeddings: Why

Only considering $N$ items, digits needed for embeddings:

$N \times d$ with embedding dimension $d$

**Toy Example:** #Param of a sequential recommender $f(u, v)$ [Wang et al. 2020]

95% of the #param

| *Item Emb. | Other Model Param. |
|---|---|
| 1,280,000 | approx. 70,000 |

*10,000 items with $d = 128$

**Real Example:** Industries are dealing with billion-scale item sets! Examples include Pinterest [Eksombatchai et al. 2018] and Alibaba [Wang et al. 2018].

10 million ($\frac{1}{100}$ billion) items with $d = 128$: 128,000,000 digits

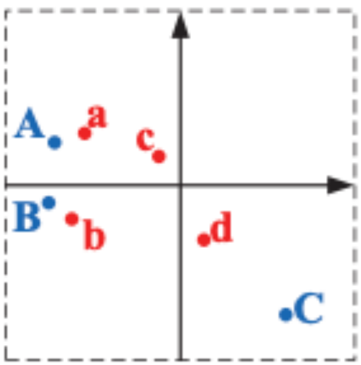
5 GB in a 32-bit system

Imagine a shopping mobile App 5-10 GB in size, and in memory!

# Binary Code-based Methods

**Early Approach – Binary Codes:**
Turn real-valued vectors into binary codes [Zhou et al. 2012, Zhang et al. 2016]

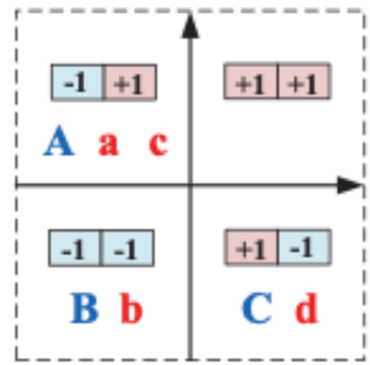
Let $\mathbf{e} \in \mathbb{R}^d$ denote a user/item embedding, and let $\mathbf{b} \in \{-1, +1\}^d$ denote the corresponding binary code




Continuous space

$$\mathbf{b} = sign(\mathbf{e})$$

$$b_{\mathrm{i}} = sign(e_{\mathrm{i}}) = \begin{cases} +1, if\ ei \geq 0 \\ -1, if\ ei < 0 \end{cases}$$




Hamming space

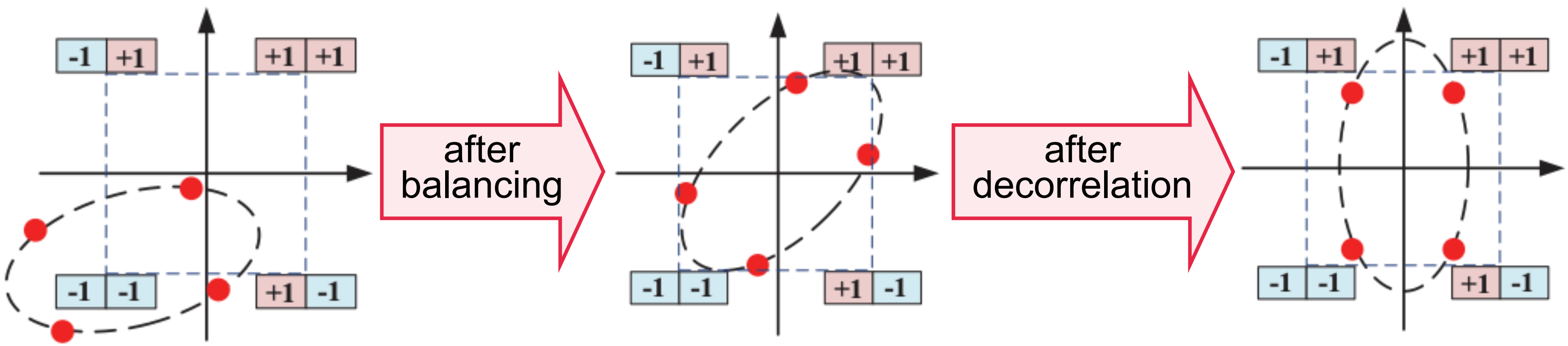
**Result:** A $d$-length code can represent $2^d - 1$ users/items (theoretically); $d = 32$ is good for 4 billion.
Fast similarity evaluation via logical operators; compact Boolean storage!

# Binary Codes: DCF

**Discrete Collaborative Filtering (DCF)** [Zhang et al. 2016]

$$\underset{\mathbf{B},\mathbf{D}}{\operatorname{argmin}} \sum_{i,j \in \mathcal{V}} \left( S_{ij} - \mathbf{b}_i^T \mathbf{d}_j \right)^2, \quad s.t.\ \mathbf{B} \in \{\pm 1\}^{r \times m}, \mathbf{D} \in \{\pm 1\}^{r \times n}$$

$$\underbrace{\mathbf{B1} = 0, \mathbf{D1} = 0,}_{\text{Balanced Partition}} \ \underbrace{\mathbf{BB}^T = m\mathbf{I}, \mathbf{DD}^T = n\mathbf{I}}_{\text{Decorrelation}}.$$




**Improvements compared with earlier variants:**

Less squashing, and binary codes are more mutually discriminative

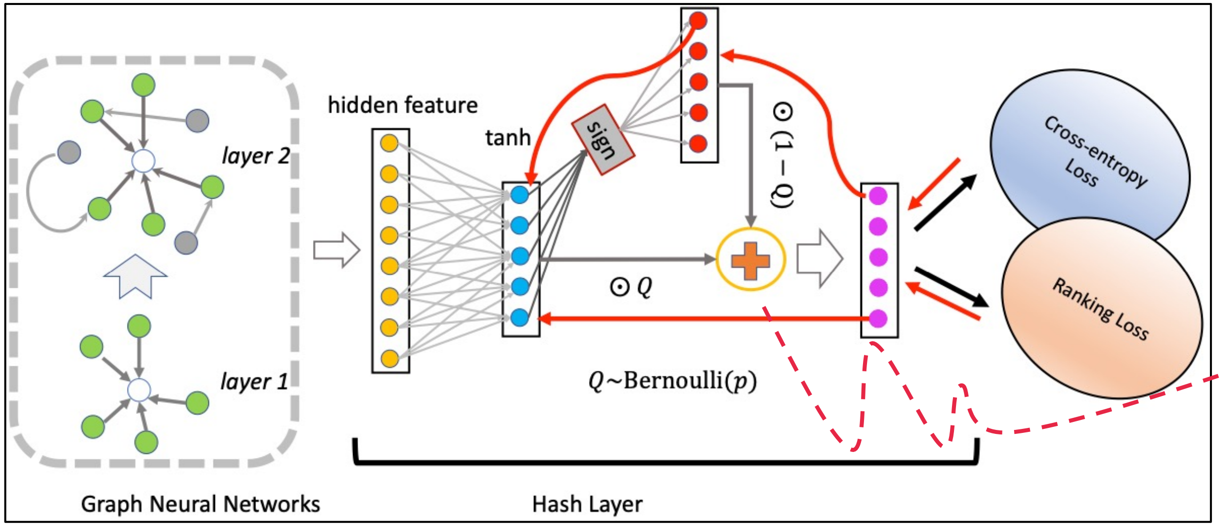


Paper: Zhang et al., "Discrete collaborative filtering", SIGIR 2016

# Binary Codes: HashGNN

**Deep Hashing with GNNs (HashGNN)** [Tan et al. 2020]

Straight-through estimator (STE) [Bengio et al. 2013] is used for back propagation.




Gating between real-valued ($\mathbf{z}$) and binary ($\mathbf{h}$) representations:

$$\hat{\mathbf{h}}_i = Q \odot \mathbf{z}_i + (1-Q) \odot \mathbf{h}_i$$

Two joint losses are minimized for higher-quality binary codes

$$\mathcal{L}_{cross} = -\sum_{\mathbf{A}_{ij} \in \mathbf{A}} \mathbf{A}_{ij} \log(\sigma(\langle \mathbf{h}_i, \mathbf{h}_j \rangle)) + (1-\mathbf{A}_{ij}) \log(1-\sigma(\langle \mathbf{h}_i, \mathbf{h}_j \rangle))$$ Reconstruct observed interactions

$$\mathcal{L}_{rank} = \sum_{(v_i, v_j, v_m) \in \mathcal{D}} \max(0, -\sigma(\langle \mathbf{h}_i, \mathbf{h}_j \rangle) + \sigma(\langle \mathbf{h}_i, \mathbf{h}_m \rangle) + \alpha)$$ BPR loss with negative samples

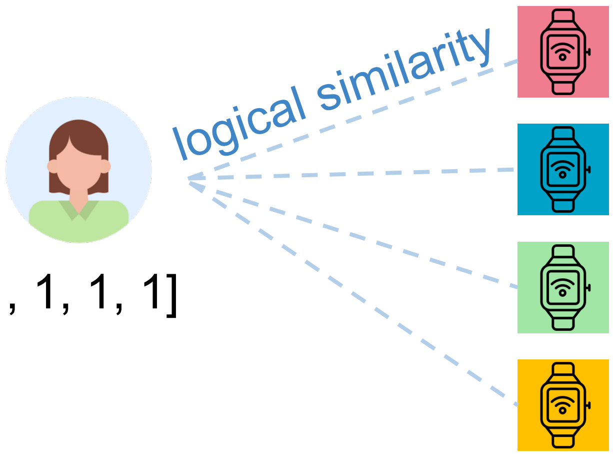


Paper: Tan et al., "Learning to hash with graph neural networks for recommender systems", WWW 2020

# Cons of Binary Codes

**Distinctiveness of individual binary codes** ≠ **Informative similarity score produced by** $f(u, v)$

**Example:**




[1, 1, 1, 1]

[-1, 1, 1, 1] $f(u, v_1) = 0.75$

[1,-1, 1, 1] $f(u, v_2) = 0.75$

[1, 1,-1, 1] $f(u, v_3) = 0.75$

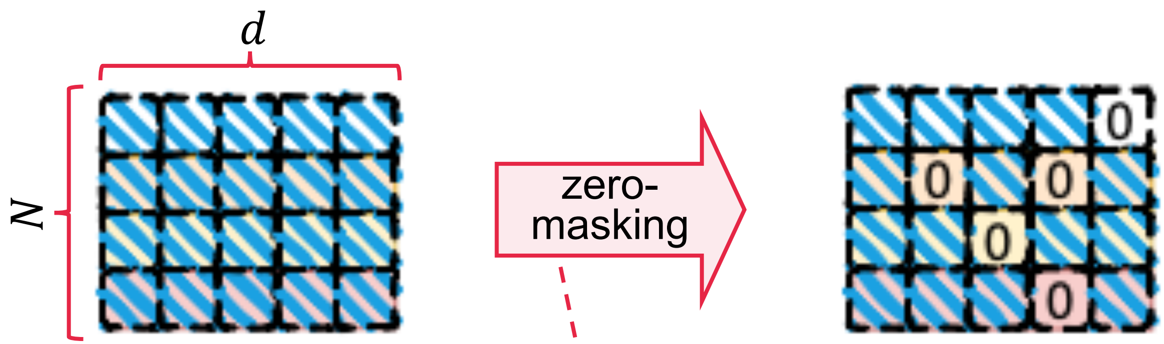
[1, 1, 1,-1] $f(u, v_4) = 0.75$

Same scores, undistinguishable for ranking!

**Result:** Binary codes have strong performance compromise.

# Embedding Sparsification Methods

So, can we shift back to **real-valued** embeddings, but make them **lighter**?




Dense embedding matrix $\mathbf{E} \in \mathbb{R}^{N \times d}$

Sparsified embedding matrix $\mathbf{E}' \in \mathbb{R}^{N \times d}$, $\|\mathbf{E}'\|_0 \leq t$ → sparsity threshold

★ Trainable algorithms (more on this later)

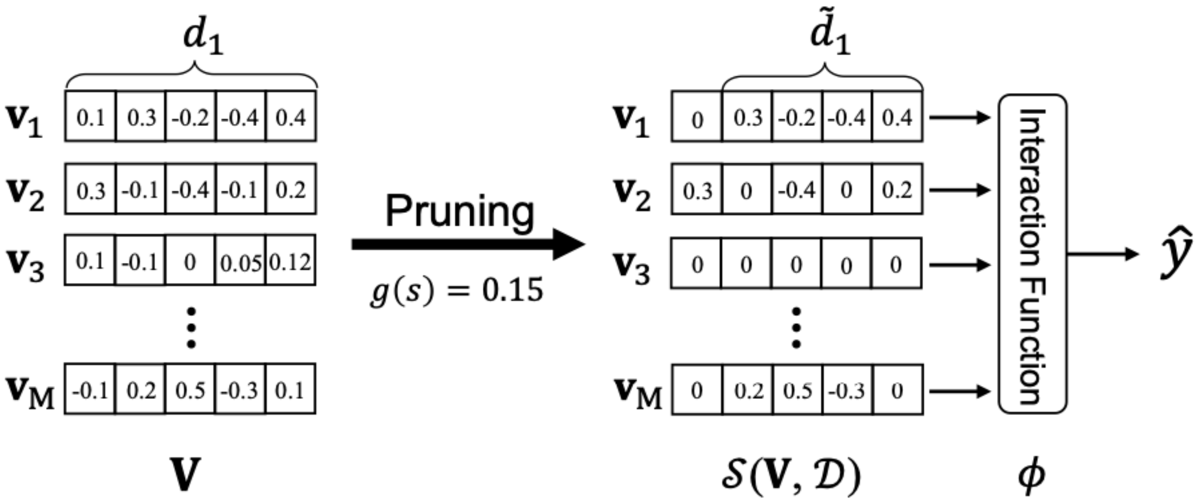
**Result:** Sparsified matrices can be **efficiently stored** [Sedaghati et al. 2015, Virtanen et al. 2020] – only $t$ matters; **consistent length** $d$ does not affect subsequent computations.

# Embedding Sparsification: PEP

**Plug-in Embedding Pruning (PEP)** [Liu et al. 2021]

Directly selecting $t$ entries to keep while minimizing $\mathcal{L}_{rec}$ is NP-hard

Can we learn it? What if we don't want to use reinforce learning?




Reparameterization: $\hat{\mathbf{V}} = \mathcal{S}(\mathbf{V}, s) = sign(\mathbf{V})\text{ReLU}(|\mathbf{V}| - g(s))$

$s$: learnable
$g(\cdot)$: sigmoid

A large $g(s)$ drops out the corresponding entry in $\mathbf{V}$ due to the effect of ReLU

After $s$ achieves the sparsity, stop pruning and retrain the sparsified RS



Paper: Liu et al., "Learnable embedding sizes for recommender systems", ICLR 2021

# Embedding Sparsification: SSEDS

**Single-Shot Embedding Dimension Search (SSEDS)** [Qu et al. 2022]

Can we do this quicker than PEP, e.g., in one shot?

$$\min_{\hat{\mathbf{V}},\hat{\Theta}}, \mathcal{L}(\hat{\mathbf{V}} \odot \boldsymbol{\alpha}, \hat{\Theta}; \mathcal{D}) \quad s.t.\ \boldsymbol{\alpha} \in \{0,1\}^{d \times \sum_i^m n_i},\ \|\boldsymbol{\alpha}\|_0 < \kappa \|\mathbf{V}\|_0$$

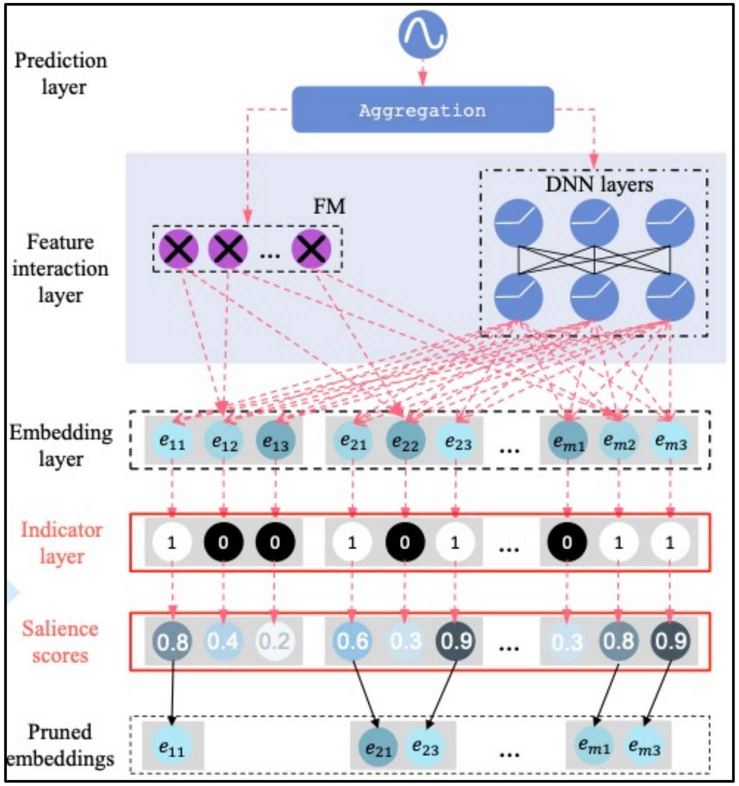
To decide the binary mask $\boldsymbol{\alpha}$, it comes down to the importance of each dimension in $d$:

$$\Delta\mathcal{L}_{i,j} = \mathcal{L}(\hat{\mathbf{V}} \odot \mathbf{1}, \hat{\Theta}; \mathcal{D}) - \mathcal{L}(\hat{\mathbf{V}} \odot (\mathbf{1} - \boldsymbol{\epsilon}_{i,j}), \hat{\Theta}; \mathcal{D})$$

Speed up with continuous relaxation of $\boldsymbol{\alpha}$:

$$\Delta\mathcal{L}_{i,j} \approx g_{i,j}(\hat{\mathbf{V}}, \hat{\Theta}; \mathcal{D}_b) = \left.\frac{\partial \mathcal{L}(\hat{\mathbf{V}} \odot \boldsymbol{\alpha}, \hat{\Theta}; \mathcal{D}_b)}{\partial \boldsymbol{\alpha}_{i,j}}\right|_{\boldsymbol{\alpha}=1}$$

Given a sparsity target, prune the least important (small $g_{i,j}$ magnitude) entries from the embedding table until target is met

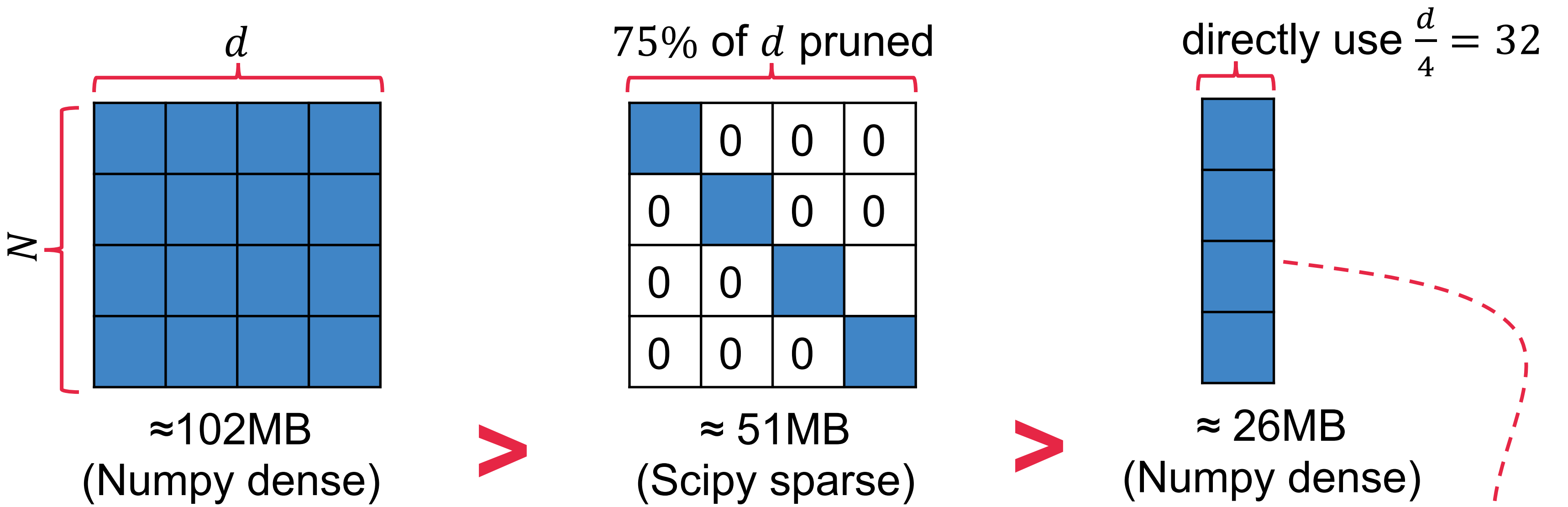





Paper: Qu et al., "Single-shot embedding dimension search in recommender system", SIGIR 2022

# Cons of Embedding Sparsification

A quick memory test with $N = 100{,}000$, $d = 128$:




Sparsification needs **extra parameters** [Lyu et al. 2022] to index 0s

Not as good as $N \times \frac{d}{4}$ dense!

**Will this be a solution?**

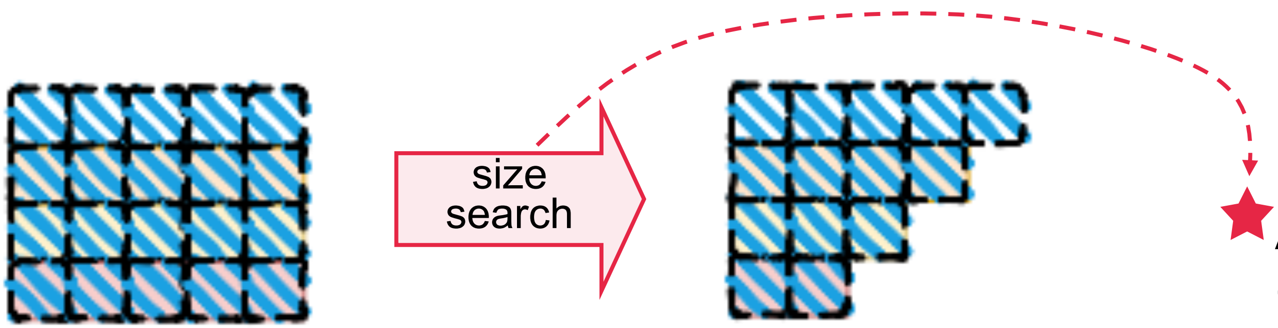
# Variable Size Embedding Methods

**Variable Size Embeddings can address the sparsification dilemma**

Should we simply shrink $d$ to $d' \ll d$ uniformly? So $N \times d' \ll N \times d$.

**Intuition:** Still taking item as an example – each has **different importance** to the recommendation task, hence does not require equal embedding dimensions.

**Relaxation:** We allow each item $v$'s embedding size $d_v$ to vary, as long as we result in $N \times d'$ total parameters!




AutoML algorithms (more on this later)

Dense embedding matrix $\mathbf{E} \in \mathbb{R}^{N \times d}$

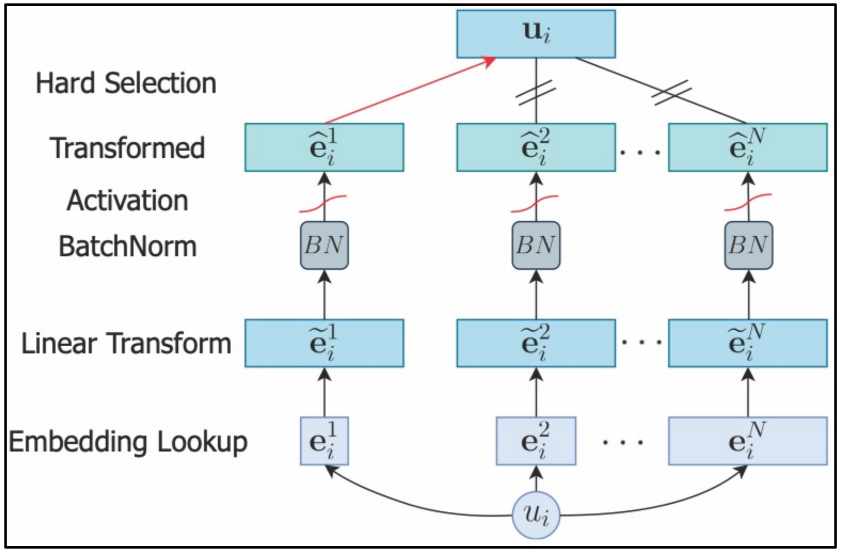
Variable size embeddings, $\sum_{v=1}^{N} d_v \leq t$

# Variable Size Embeddings: AutoEmb

**Optimal Embedding Table Learning (AutoEmb)** [Zhao et al. 2021]

**In AutoML:** Try different candidate neural nets, pick best one per task

**In AutoEmb:** Try different candidate $d_v \in \{2,8,32,64\}$, pick best one per $v$




Two optimization pathways:

- Reinforcement learning (RL, hard pick [Joglekar et al. 2020])
- Differentiable search (DARTS [Liu et al. 2018])

$$\min_{\Theta} \mathcal{L}_{val}(\Psi^*(\Theta), \Theta) \quad s.t.\ \Psi^*(\Theta) = \arg\min_{\Psi} \mathcal{L}_{train}(\Psi, \Theta^*)$$

$\mathcal{L}$ can be performance-oriented or incorporate size/diversity constraints.



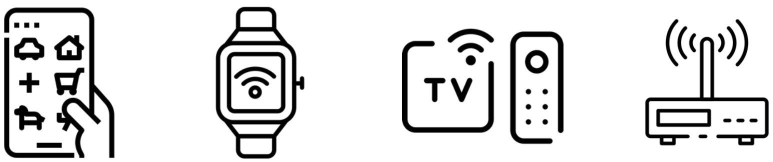
Paper: Zhao et al., "Autoemb: Automated embedding dimensionality search in streaming recommendations", ICDM 2021

# Possible Directions for Improvement

**Note 1: Real-world practicality** of variable size embeddings

No one-size-fits-all solutions!

**Category-wise heterogeneity**



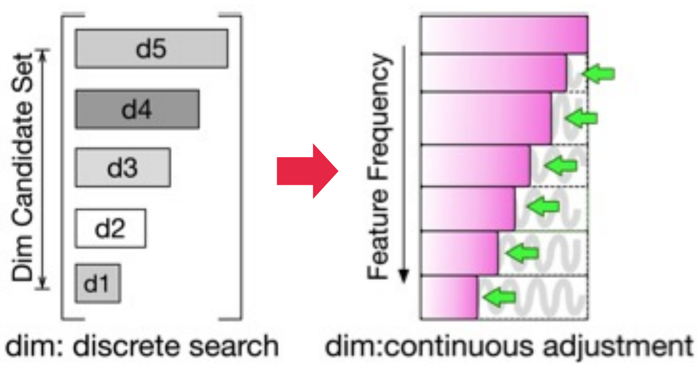

**Age-wise heterogeneity**

iPhone 8 (2017): 2 GB RAM

iPhone 12 Pro (2020): 6 GB RAM

**Note 2: Performance bottleneck** of variable size embeddings

Coarse-grained search candidates, e.g., $d_v \in \{2,8,32,64\}$, too discrete!

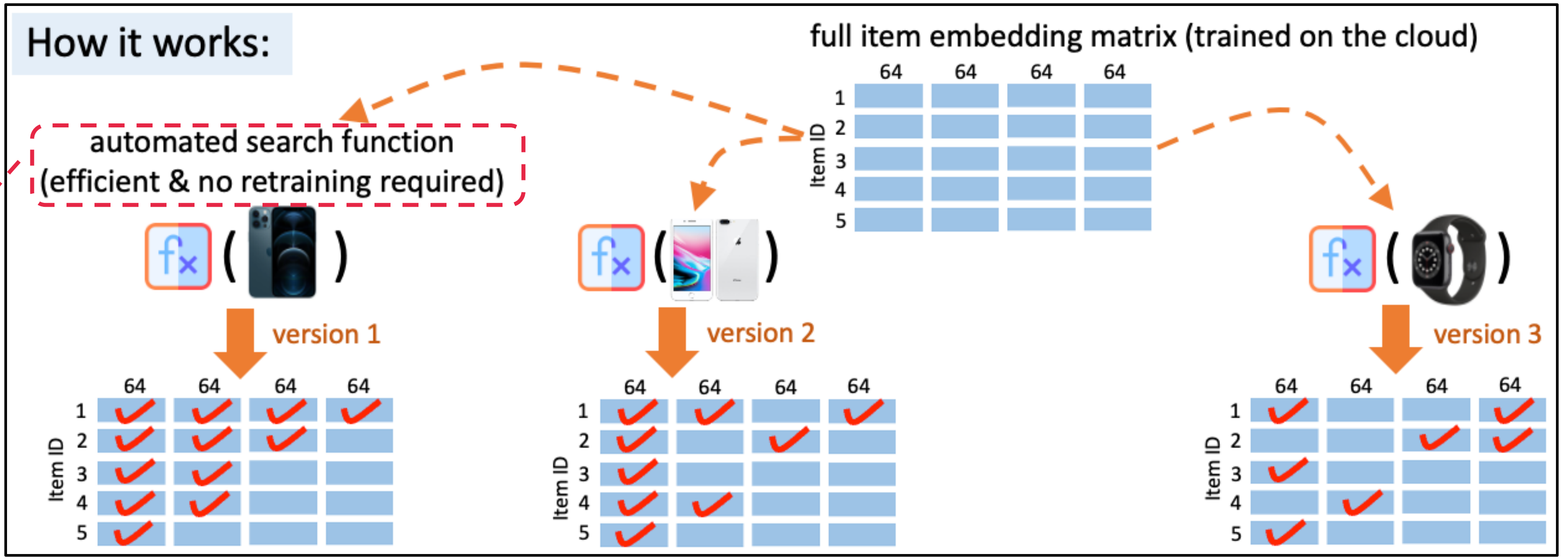



[Yan et al. 2021]

Using (near) continuous search interval, e.g., $d_v \in [1,128] \cap \mathbb{N}$ is desirable but costly!

# Variable Size Embeddings: RULE

Improvement on Note 1: **Recommendation with Universally Learned Elastic Embeddings (RULE)** [Chen et al. 2021]




**Memory-bounded evolutionary search** [Real et al. 2019] is proposed for:

- Performance (of size combinations) estimator output $\hat{y}$
- Diversity (of retained entries) regularizer $\mathcal{L}_{reg}$

A plus version: Personalized Elastic Embedding (PEE) [Zheng et al. 2024].

Papers: Zheng et al., "Personalized Elastic Embedding Learning for On-Device Recommendation", TKDE 2024
Chen et al., "Learning elastic embeddings for customizing on-device recommenders", KDD 2021

# Variable Size Embeddings: CIESS

Improvement on Note 2: **Continuous Input Embedding Size Search (CIESS)** [Qu et al. 2023]

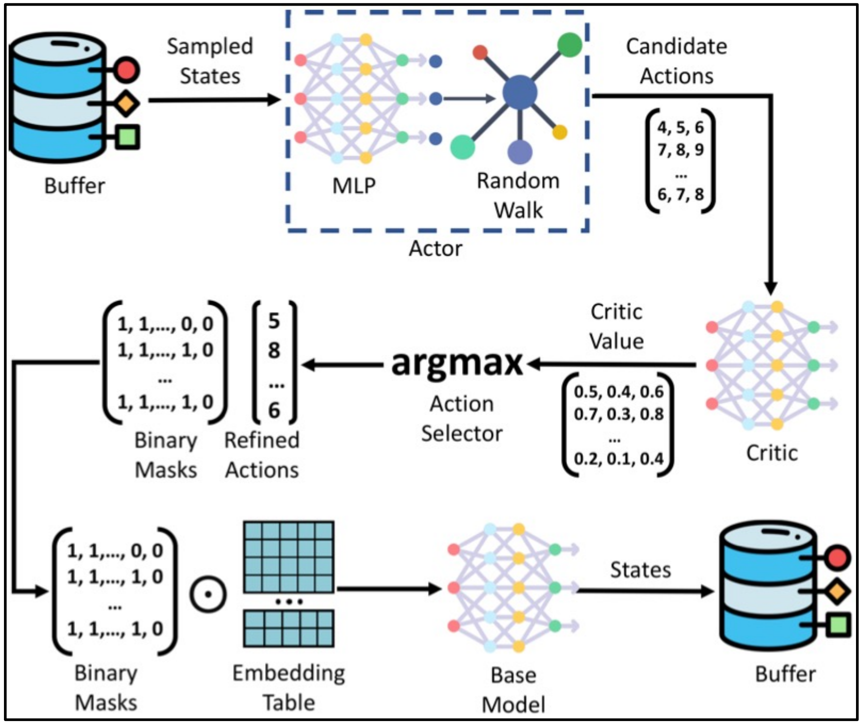
Treating all possible $d_v$ in $[1, d]$ range as discrete candidates in RL is not ideal

A more careful candidate exploration strategy - using random walk

$$p(d \rightarrow d') = \frac{|d' - d|}{\sum_{d' \in \mathcal{A}_d} |d' - d|}$$

with a noisy start

$$\hat{d}_v^i = \mu_\gamma(s_v^i) + \epsilon \quad \epsilon \sim \mathcal{N}(0, \sigma)$$




CIESS uses twin delayed deep deterministic policy gradient (TD3) [Fujimoto et al. 2018] as the RL optimizer.



Paper: Qu et al., "Continuous input embedding size search for recommender systems", SIGIR 2023

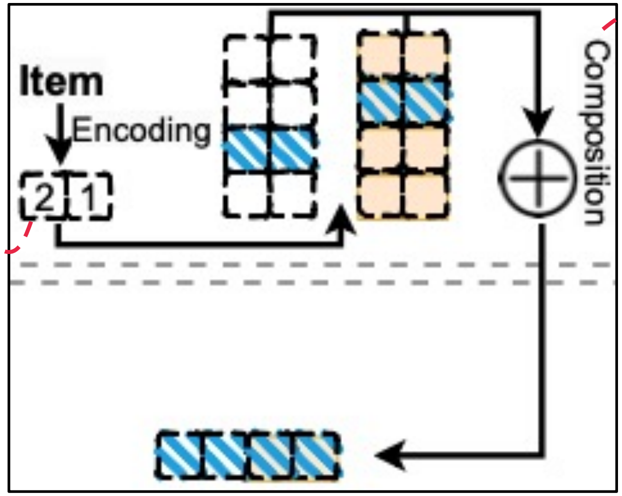
# Compositional Embedding Methods

**Compositional Embedding Methods cure the sparsification dilemma, too**

Is shrinking $d$ the only way out? When $N \times d$ becomes $N' \times d$ ($N \ll N'$), how to make sure each item gets a unique embedding?

**Intuition:** For each $v$, If we pick two (or more) embeddings and compose into one, then the $N' \times d$ matric can optimally represent $\binom{N'}{2}$ items!

★ Predefined/Learnable assignment (more on that later)




Can be concatenate, add, element-wise product, etc.

We can slice $N' \times d$ into $s$ smaller chunks (a.k.a. codebooks), as long as $(\frac{N'}{s})^s \geq N$

# Compositional Embeddings: QRT

**Quotient-Remainder Trick (QRT)** [Shi et al. 2020]

Inspired by dual-hashing, QRT uses the collision-free quotient-remainder formulation to hash each item ID into $k$ codebook indexes $P_1, P_2, \ldots Pk$.

$$P_1 = \{\{x \in S : \varepsilon(x) \bmod m_1 = l\} : l \in \mathcal{E}(m_1)\}$$

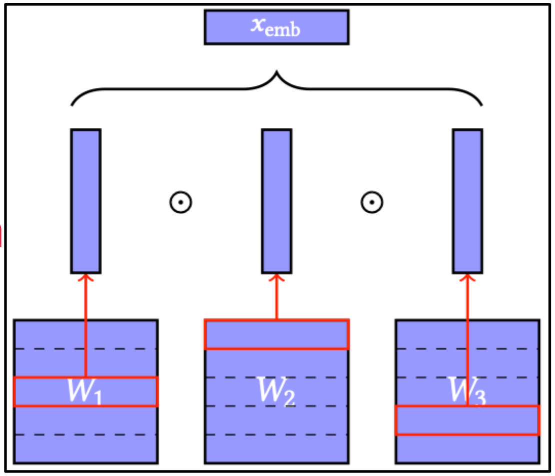
$$P_j = \{\{x \in S : \varepsilon(x) \backslash M_j \bmod m_j = l\} : l \in \mathcal{E}(m_j)\} \quad M_j = \prod_{i=1}^{j-1} m_i$$

Two variants: linear (left) and path-based (right) compositions

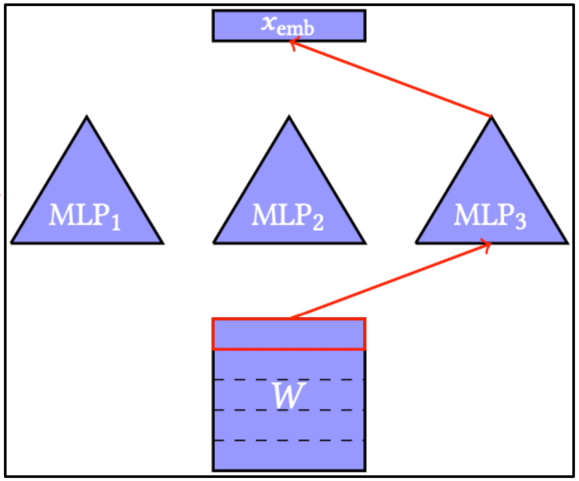
$$\boldsymbol{x}_{\text{emb}} = \omega(\boldsymbol{W}_1^T \boldsymbol{e}_{p_1(x)}, \boldsymbol{W}_2^T \boldsymbol{e}_{p_2(x)}, \ldots, \boldsymbol{W}_k^T \boldsymbol{e}_{p_k(x)})$$

$$\boldsymbol{x}_{\text{emb}} = (M_{k,p_k(x)} \circ \ldots \circ M_{2,p_2(x)})(\boldsymbol{W}\boldsymbol{e}_{p_1(x)})$$

Faster computation




More Parameter Reduction





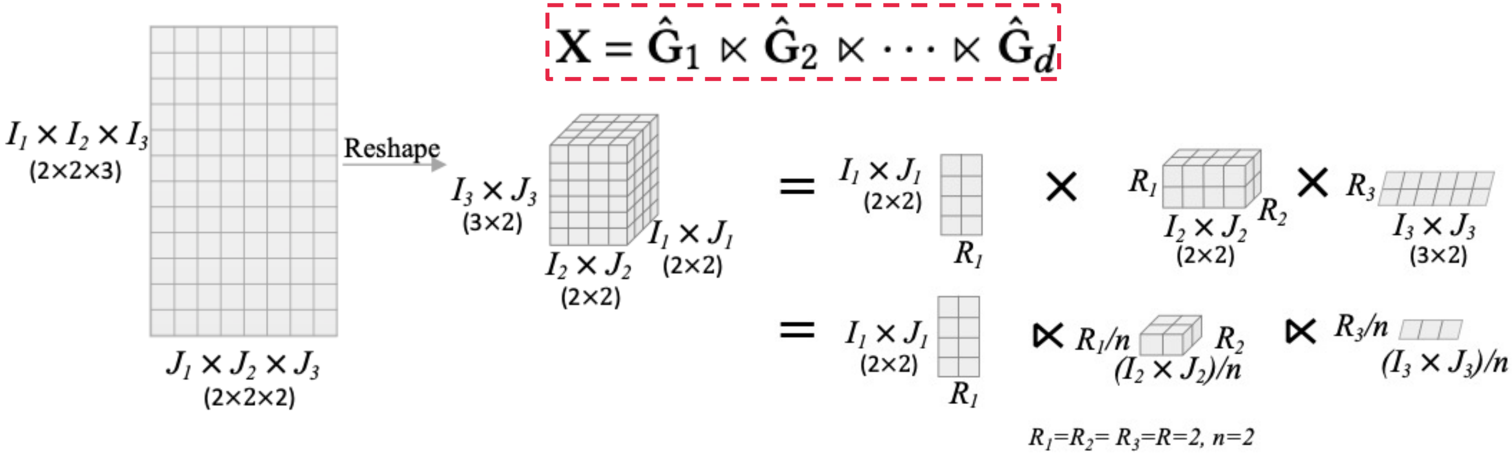

Paper: Shi et al., "Compositional embeddings using complementary partitions for memory-efficient recommendation systems", KDD 2020

# Compositional Embeddings: ODRec

**Ultra-Compact On-Device Recommendation (ODRec)** [Xia et al. 2022]

Recall matrix factorization – can we decompose the embedding matrix into a sequence of matrix (tensor) products? ODRec uses semi-tensor product (STP)!

$$\mathbf{X} = \hat{\mathbf{G}}_1 \ltimes \hat{\mathbf{G}}_2 \ltimes \cdots \ltimes \hat{\mathbf{G}}_d$$




Toy example – how STP shrinks a 12×8 embedding table into STP form with 2×2-, 1×2x2-, and 1x3-shaped tensors. Further enhancements in ODRec:

- Knowledge distillation from a full teacher model
- Contrastive learning in the on-device model



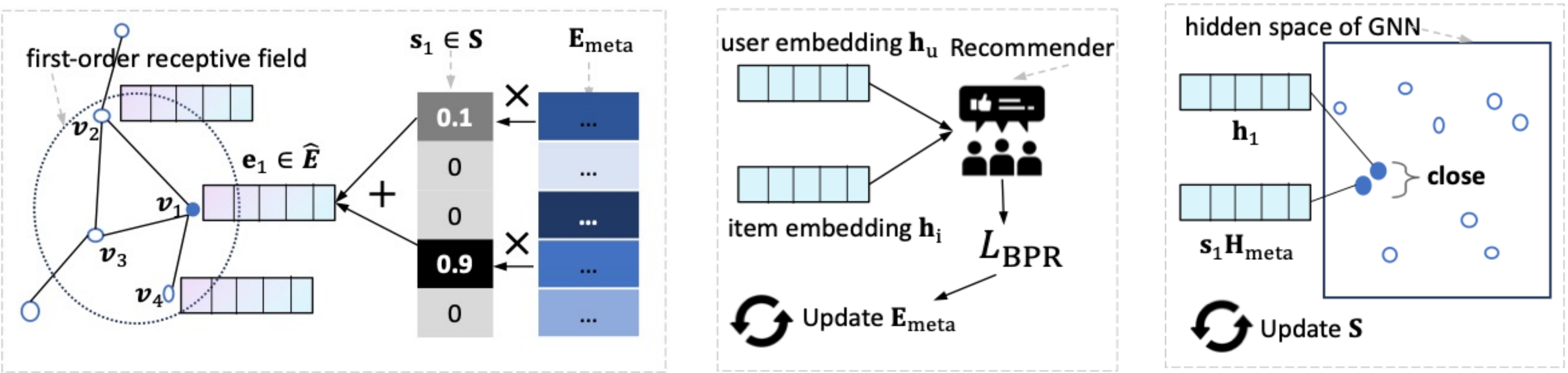
Paper: Xia et al., "On-device next-item recommendation with self-supervised knowledge distillation", SIGIR 2022

# Compositional Embeddings: LEGCF

**Lightweight Embeddings for Graph Collaborative Filtering (LEGCF)** [Liang et al. 2024]

Why predefine compositional assignment $\mathbf{S}$ when we can learn one?

$\widehat{\mathbf{E}} = \mathbf{S}\mathbf{E}_{meta}$ where $\mathbf{S} \in \mathbb{R}_{\geq 0}^{N \times N'}$ is a trainable sparse assignment matrix




The codebook $\mathbf{E}_{meta} \in \mathbb{R}^{N' \times d}$ is trained via gradient descent. To avoid co-adaptation [Hinton et al. 2012] between $\mathbf{S}$ and $\mathbf{E}_{meta}$, $\mathbf{S}$ is updated in closed form:

$$\mathbf{H}_{full} \approx \mathbf{S}\mathbf{H}_{meta} \Rightarrow \mathbf{H}_{full}\mathbf{V}\Sigma^{-1}\mathbf{U}^{*}$$



Paper: Liang et al., "Lightweight Embeddings for Graph Collaborative Filtering", SIGIR 2024

# Variable Size vs Composition

**Both** are sound solutions to the sparsification dilemma.

So, when to use which? It really depends.

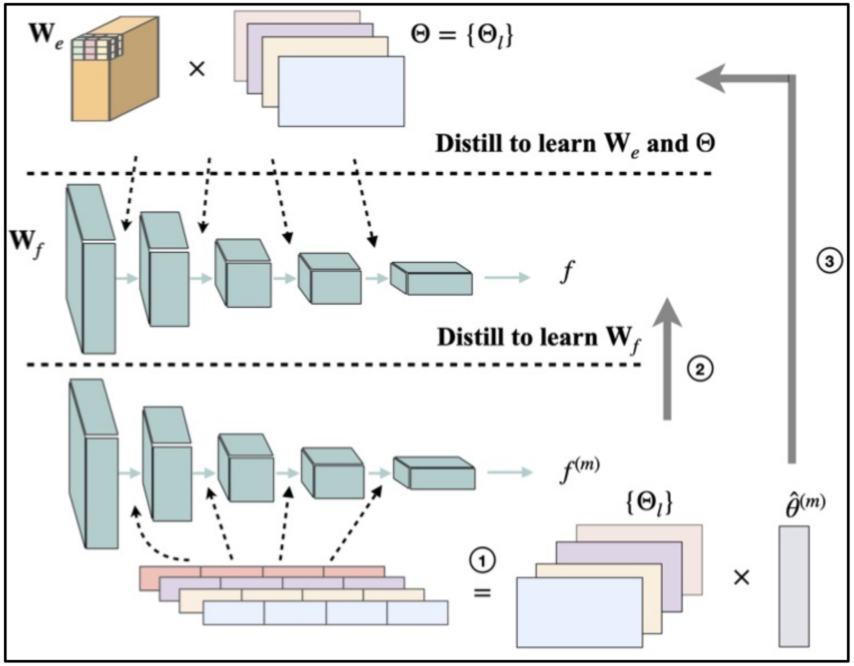
Variable Size Embeddings (VSE) vs. Compositional Embeddings (CE):

| Desiderata | VSE | CE |
|---|---|---|
| Better memory efficiency than embedding sparsification | + | + |
| Flexible embedding dimension based on importance | + | – |
| No additional assignment storage | + | – |
| No need to modify downstream similarity functions | – | + |
| New user/items | – | ? |

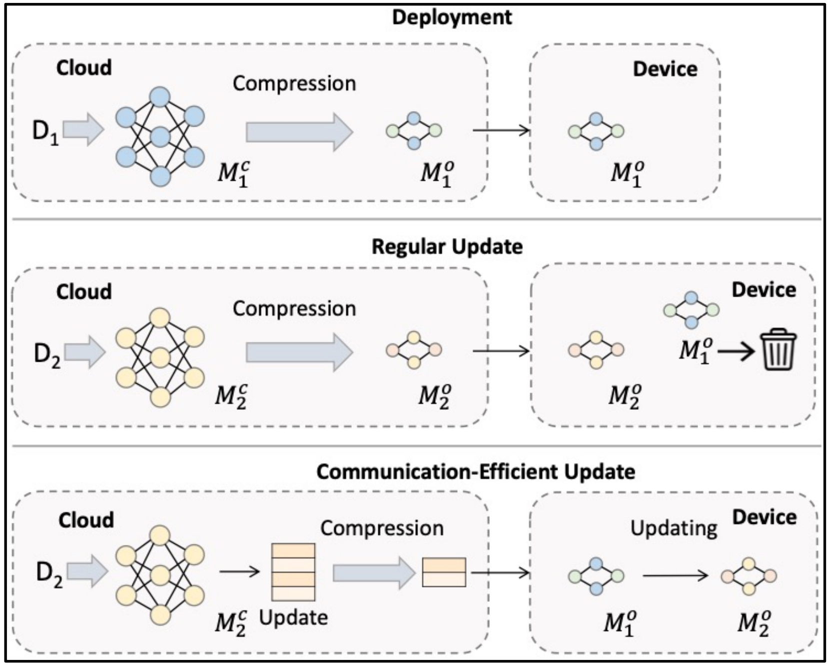
# Sustainable Deployment: Stay Up-to-date

In many cases, deployment of ODRSs are not just one-off.

To keep on-device models up-to-date, patch learning is a solid choice.




**Model-over-Models Distillation (MoMoDistill)** [Yao et al. 2021]




**Communication-Efficient On-Device Model Update (ODUpdate)** [Xia et al. 2023]



Papers: Yao et al., "Device-cloud collaborative learning for recommendation", KDD 2021
Xia et al., "Efficient on-device session-based recommendation", TOIS 2023

# Training and Updating for ODRSs

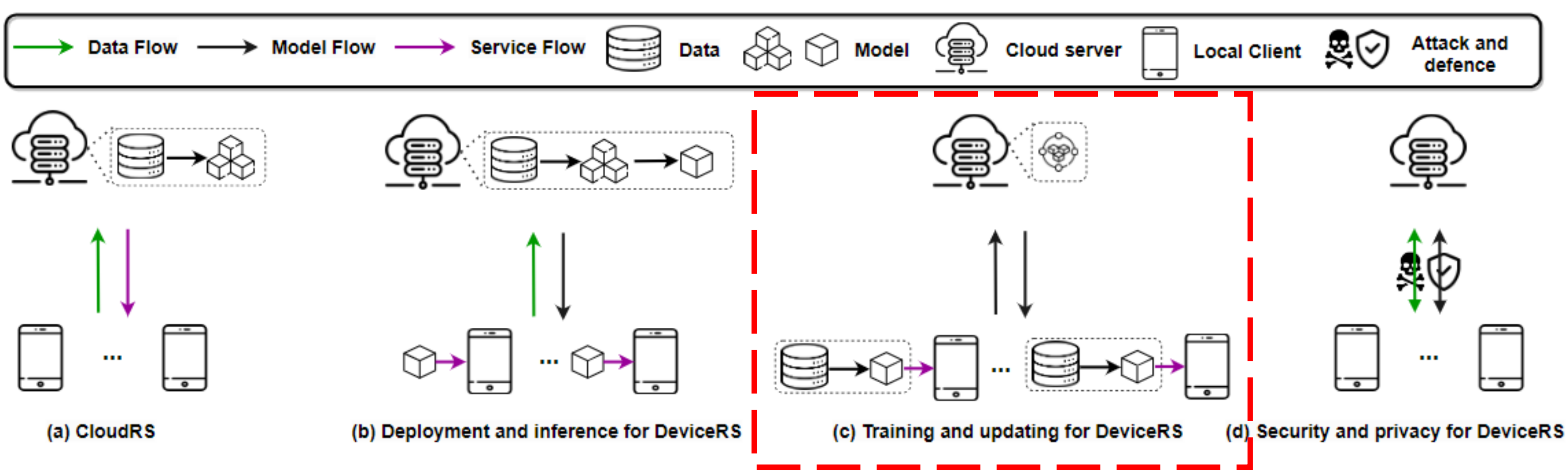
**Motivation**:

- Privacy concerns
- Real-time changes in user interests

**Challenge:** Limited user-item interaction data on devices complicates achieving high performance through local training alone.

**Solutions:** Shift parts or all of the model training and updating to the device side.

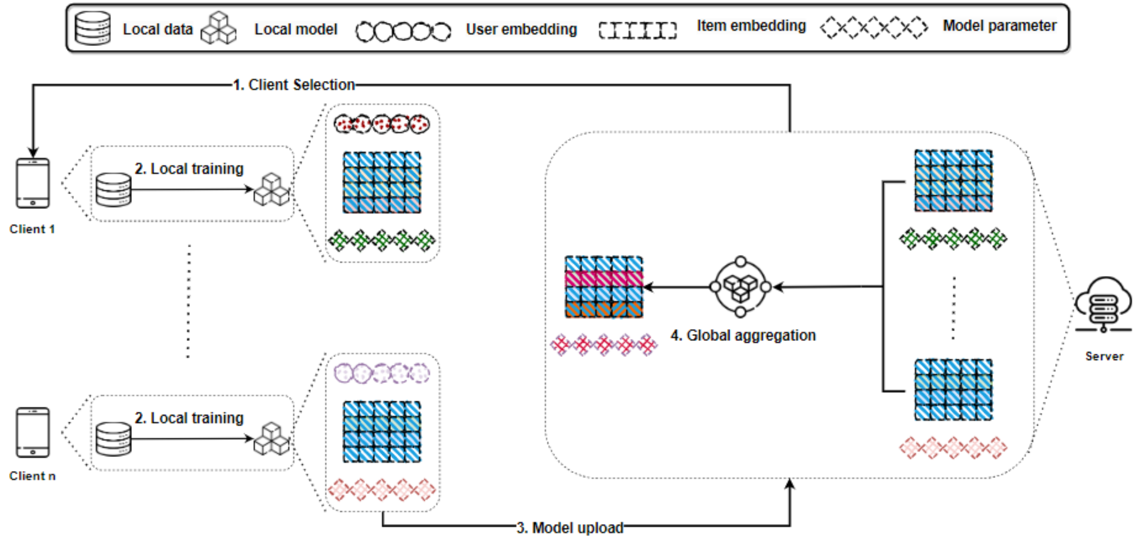
- **Federated RSs:** Enhances training through device-to-server communications.
- **Decentralized RSs:** Facilitates device-to-device collaborative training.
- **Finetuning RSs:** Utilizes local data to refine pre-trained models from the server.



# Federated Recommendation Methods

**Key Idea:** Maintains privacy by avoiding direct data sharing, focusing solely on parameter/gradients exchange.

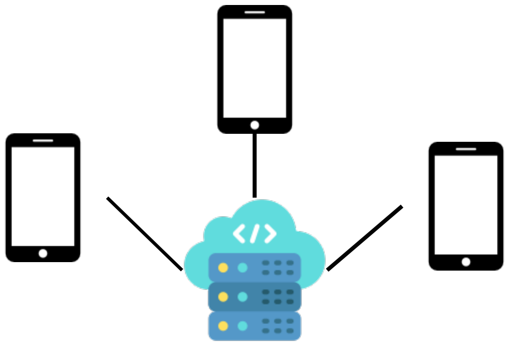
- **Client selection:** The server selects devices based on a client selection strategy.
- **Local Training**: Selected clients train models using their local datasets.
- **Model Upload**: Devices upload trained model parameters or gradients back to the server.
- **Global Aggregation**: Server aggregates received parameters to update the global model.



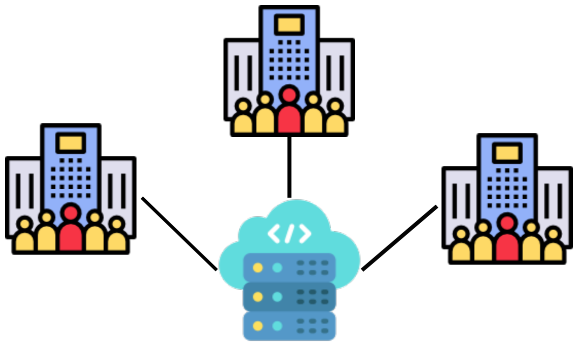
# Client Selection

**Purpose**: To select a subset of clients for participation in each training round.
**Challenge**: Client data is often non-iid, making the selection process crucial for model performance.
**Selection Strategies**:

- **Random Selection**:
  - Clients are chosen randomly for each training round.
- **Full Selection**:
  - All clients participate in every training round.
- **Clustering-based Selection**:
  - Group clients into clusters based on similarity to enhance the efficiency and effectiveness of the training process.



**Cross-client FedRSs**



**Cross-platform FedRSs**

# Clustering-based Selection

**FedFast** [Muhammad et al. 2020]: Uses k-means for clustering by metadata or embeddings, selects one client per group.
**PerFedRS** [Luo et al. 2022]: Clusters users by uploaded embeddings and selects clients proportionally within clusters.
**SemiDFEGL** [Qu et al. 2023]: Considers both client and item embeddings during clustering and employs fuzzy c-means to allow items to overlap across different groups.

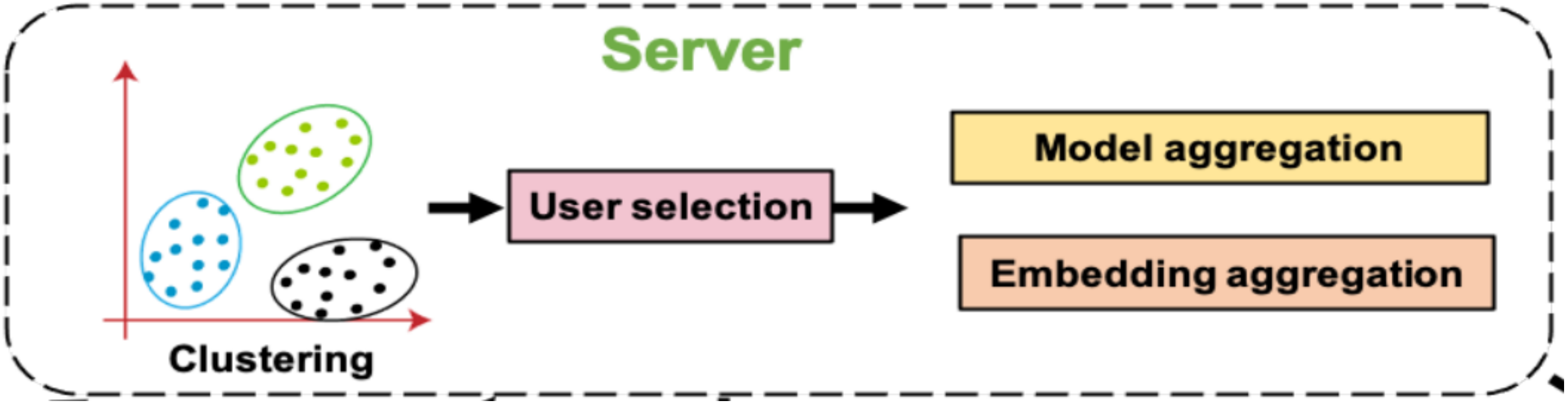
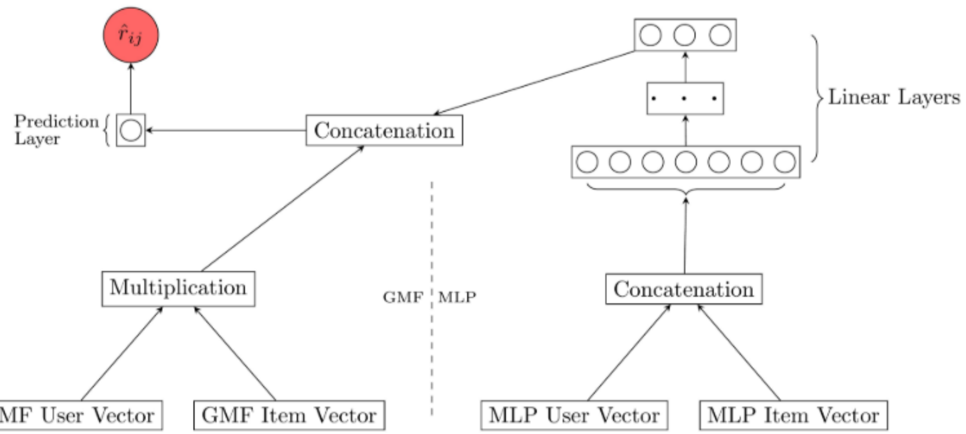



**Illustration of the proposed personalized federated recommendation framework** [Luo et al. 2022]

Papers: Qu et al., "Semi-Decentralized Federated Ego Graph Learning for Recommendation". WWW 2023
Luo et al.. "Personalized Federated Recommendation via Joint Representation Learning, User Clustering, and Model Adaptation". CIKM 2022
Muhammad et al. "FedFast: Going Beyond Average for Faster Training of Federated Recommender Systems". KDD 2020.

# Local Training and Model Upload

**NCF-Based Methods:**
The input to an NCF consists mainly of user embeddings and item embeddings
**GMF side:** uses the Hadamard product to calculate the dimension-wise interaction embeddings of users and item
**MLP side**: user and item embeddings are concatenated to serve as the input layer for the MLP




**Architecture of neural collaborative filtering (NCF)** [Perifanis et al. 2022]



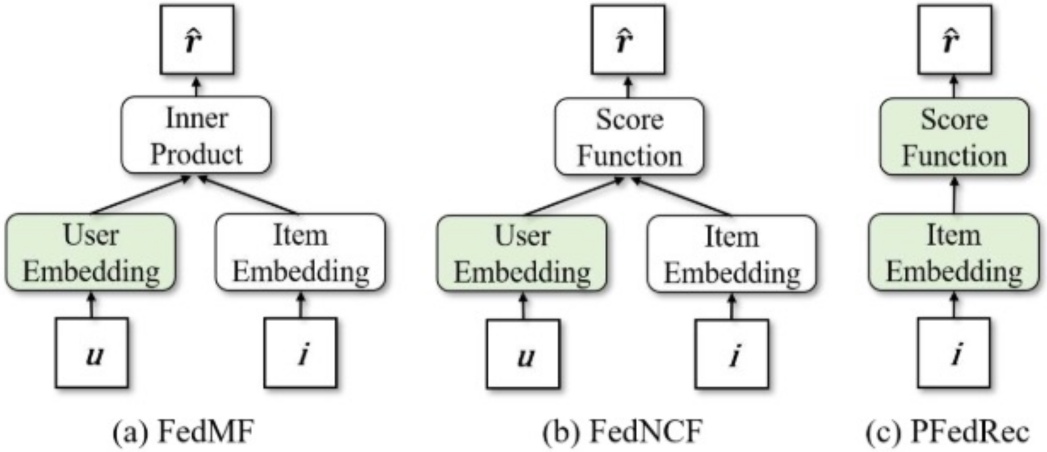
Image: Vasileios Perifanis et al., “Federated Neural Collaborative Filtering”. Know.-Based Syst. 2022,

# Local Training and Model Upload

**FedMF** [Chai et al. 2020]: Adapts matrix factorization to federated settings, updating user embeddings locally and aggregating item gradients globally to preserve privacy.

**FedNCF** [Perifanis et al. 2022]: Extends Neural Collaborative Filtering to federated environments, updating user embeddings locally and aggregating item and score function parameters globally.

**PFedRec** [Zhang et al. 2023]: Introduces dual personalization in federated recommendations, personalizing item embeddings and score functions on devices for enhanced user-specific recommendations.




Different frameworks for the federated recommendation [Zhang et al. 2023]

Papers: Chai et al., "Secure federated matrix factorization". IEEE Intelligent Systems, 2020.

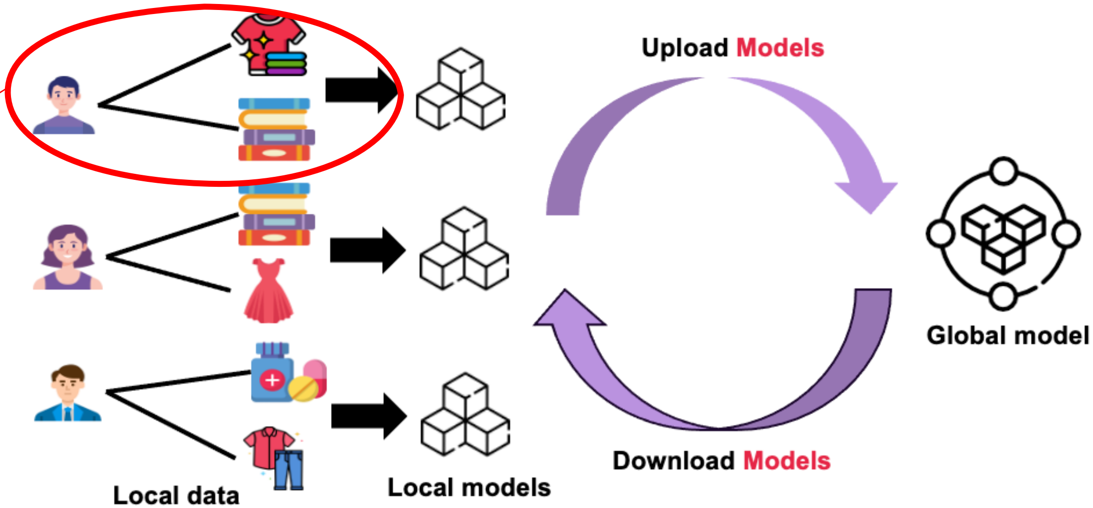
Perifanis et al., "Federated Neural Collaborative Filtering". Know.-Based Syst. 2022

Zhang et al., "Dual Personalization on Federated Recommendation". IJCAI 2023

# Local Training and Model Upload

- **Why do Federated GNN-based Recommender Systems undergo performance degradation?**
  - Local data are limited to user-centric ego graphs
  - Higher-order graph structural information cannot be directly utilized, leading to decreased model performance.

Local data are limited to ego graphs





**How can we leverage high-order graph structural information to improve model performance while maintaining privacy?**

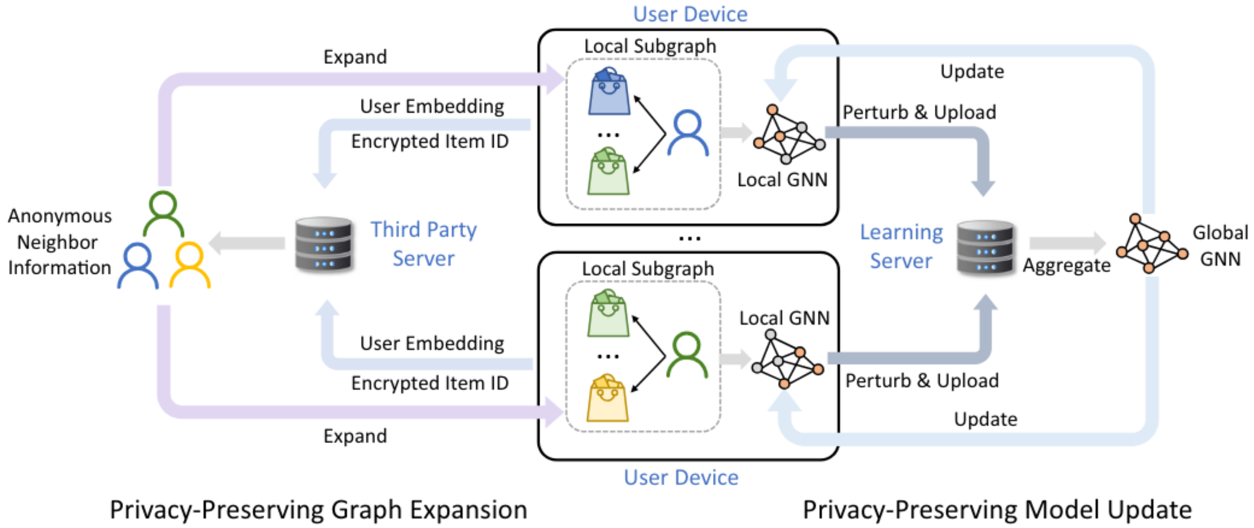


Paper: Wu et al. "A federated graph neural network framework for privacy-preserving personalization". Nat Commun. 2022.

# Local Training and Model Upload

**FedGNN** [Wu et al. 2022] uses an additional third-party server to construct the global graph.

- Users uploaded their encrypted interaction data to the third-party server.
- The third-party server constructs a global model, and shares encrypted anonymized neighbor information with clients for local training
- The encryption process introduces significant computational and communication overhead




**Architecture of FedGNN** [Wu et al. 2022]

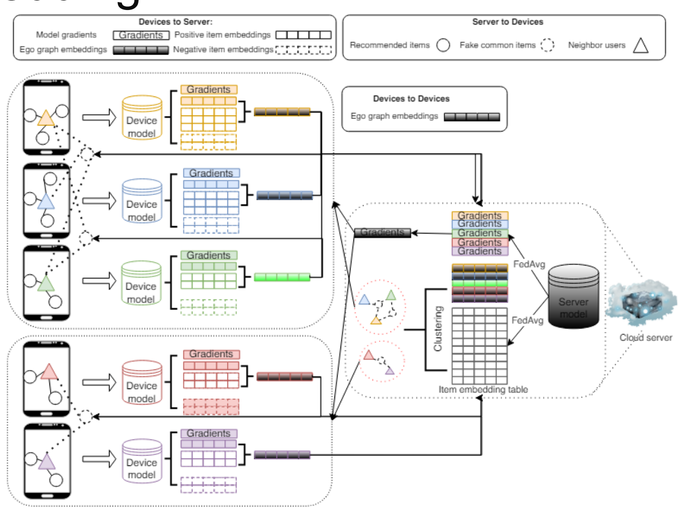


Paper: Wu et al. "A federated graph neural network framework for privacy-preserving personalization". Nat Commun. 2022.

# Local Training and Model Upload

**SemiDFEGL** [Qu et al. 2023] : Fake common items are generated to connect the isolated ego graphs of each client, thereby establishing higher-order local subgraphs.

- Each client trains locally to learn local ego-graph embedding
- Server uses ego-graph embedding and item embeddings to perform fuzzy c-means. Items grouped with a user serve as fake common neighbors to connect different users.
- Users in each group can use the generated common neighbors as a bridge to transmit user embedding.



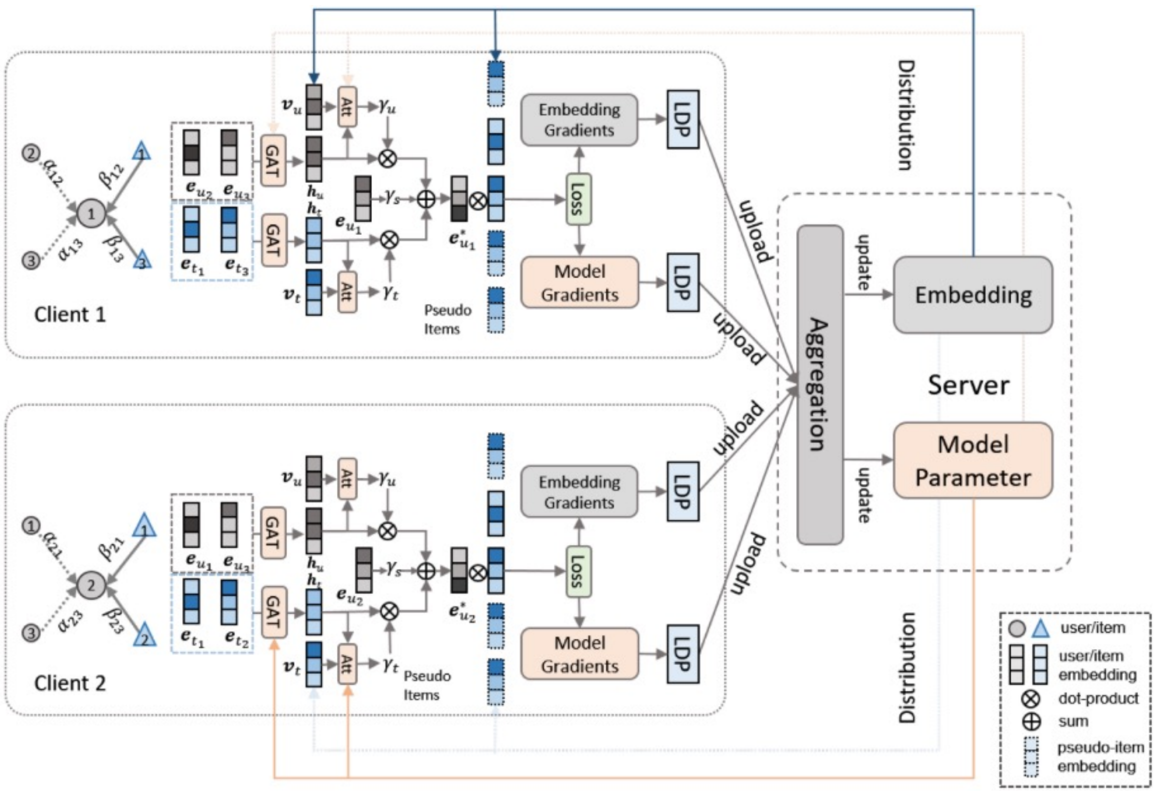

**Architecture of SemiDFEGL** [Qu et al. 2021]



Paper: Qu et al., “Semi-Decentralized Federated Ego Graph Learning for Recommendation”. WWW. 2023.

# Local Training and Model Upload

**FeSoG** [Liu et al. 2022] utilizes additional social information between users (i.e., user-user connections) to alleviate data sparsity and cold-start issues in the user-item bipartite graph.




**Architecture of FeSoG** [Liu et al. 2022]



Paper: Liu et al.,"Federated Social Recommendation with Graph Neural Network". ACM Trans. Intell. Syst. Technol. 2022

# Global Aggregation

**Key idea:** After the model upload process, the server aggregates these parameters or gradients to learn a global model.

- Gradient Descent:

$$\mathbf{Q}_s \leftarrow \mathbf{Q}_s - \gamma \sum_{u \in \mathcal{U}^+} \frac{\partial \mathcal{L}_u}{\partial \mathbf{Q}_u}$$

- FedAvg:

$$\Theta_s^{(t+1)} \leftarrow \sum_{u \in \mathcal{U}^+} \frac{|\mathcal{D}_u|}{|\mathcal{D}|} \Theta_u^{(t)}$$

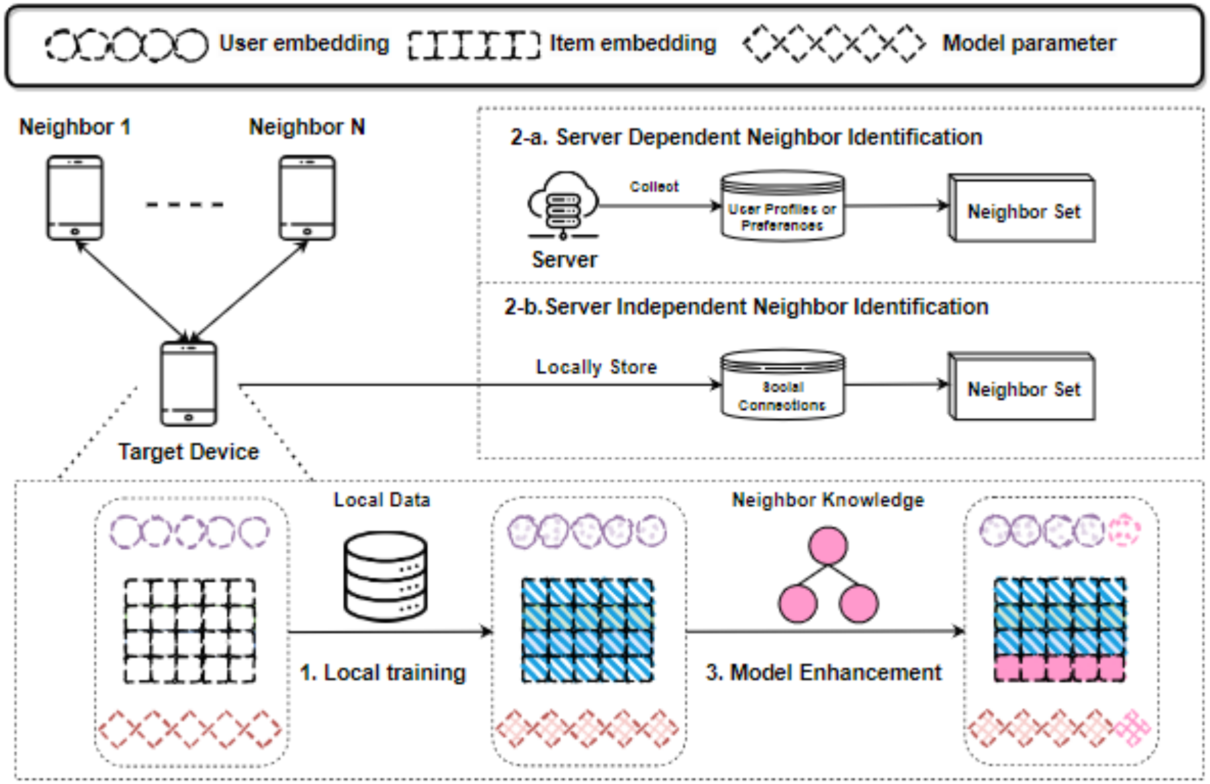
- Average Aggregation:

$$\Theta_s^{(t+1)} \leftarrow \sum_{u \in \mathcal{U}^+} \frac{1}{|\mathcal{U}^+|} \Theta_u^{(t)}$$

# Decentralized Recommendation Methods

**Motivation:** FedRSs heavily rely on a central server for aggregating user models and redistributing the combined model

**DecRSs:** DecRSs reduce reliance on central servers by optimizing models through local training and direct communication among specific user groups.

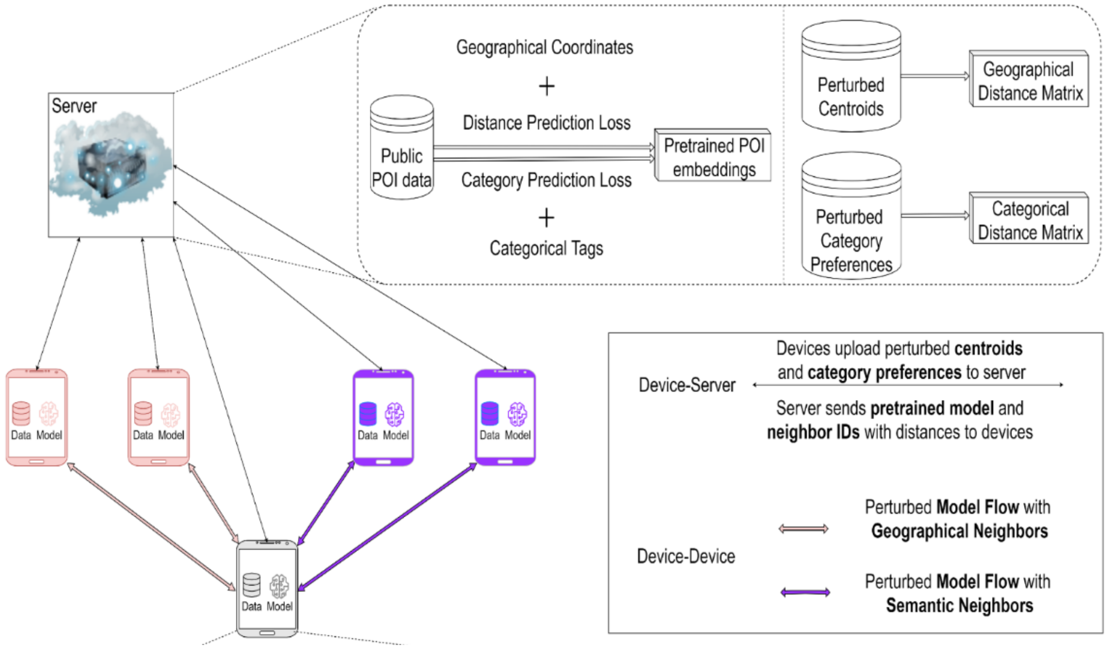



**How to choose neighbors? How do neighbors collaborate in learning?"**

# Decentralized Recommendation Methods

**DCLR** [Long et al. 2023] propose to incorporate knowledge from either geographically or semantically similar users into each local model with attentive aggregation and mutual information maximization

- Neighbor selection: geographical information and category preferences
- Collaborative learning: FedAvg




**Architecture of DCLR** [Long et al. 2023]



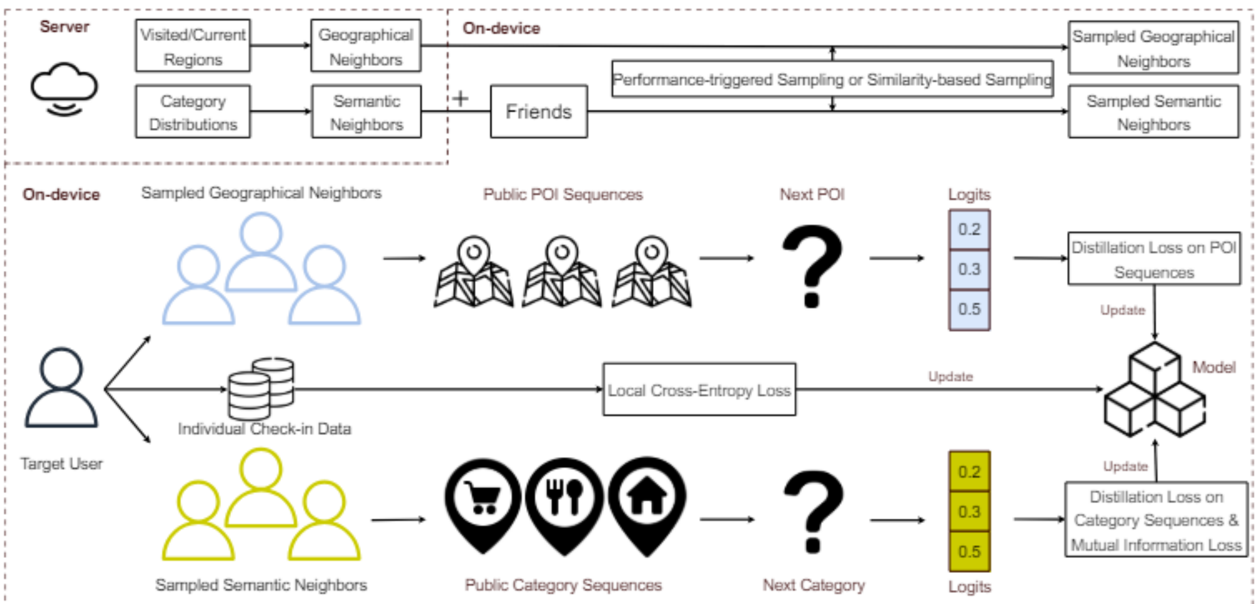
Paper: Long et al. "Decentralized Collaborative Learning Framework for Next POI Recommendation". TOIS. 2023

# Decentralized Recommendation Methods

**MAC** [Long et al. 2023] proposes a model-agnostic decentralized collaborative learning method for devices with heterogeneous models.

Neighbor selection: geographical information and category preferences

Collaborative learning: knowledge distillation




**Architecture of MAC** [Long et al. 2023]



Paper: Long et al. “Model-Agnostic Decentralized Collaborative Learning for On-Device POI Recommendation.” SIGIR. 2023.

# On-device Recommender Finetuning

FedRSs and DecRSs require substantial device computation and can lead to extensive training times, which may deter user participation.
Devices fine-tune the global model using local data to better match individual user preferences.

- **Reduced Training Demand**: Lessens the computational load on individual devices compared to full model training in FedRSs and DecRSs.
- **Increased User Engagement**: Shortens training time, potentially increasing participation from less active users.

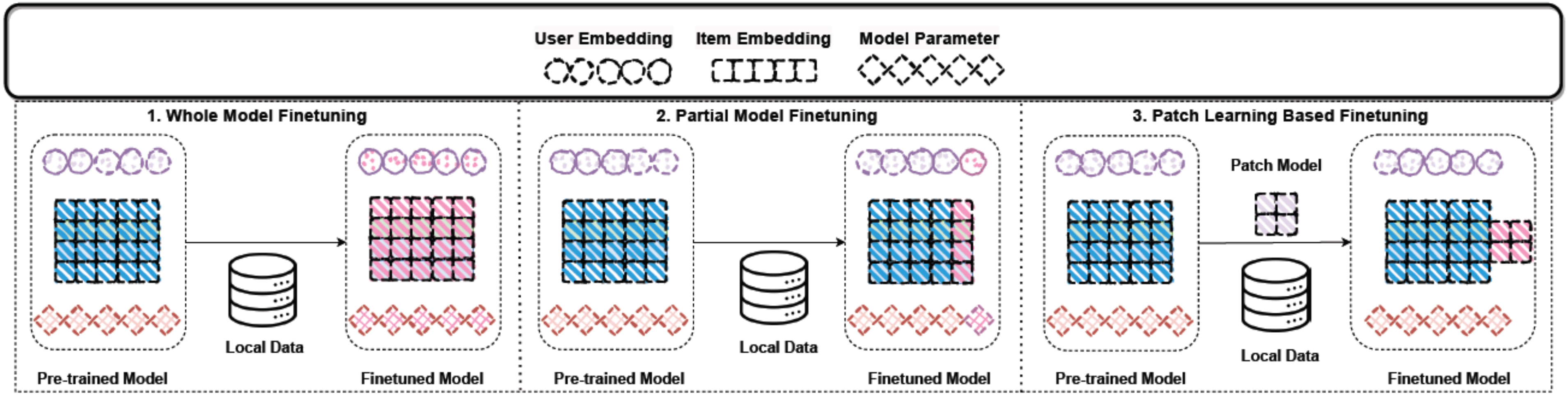
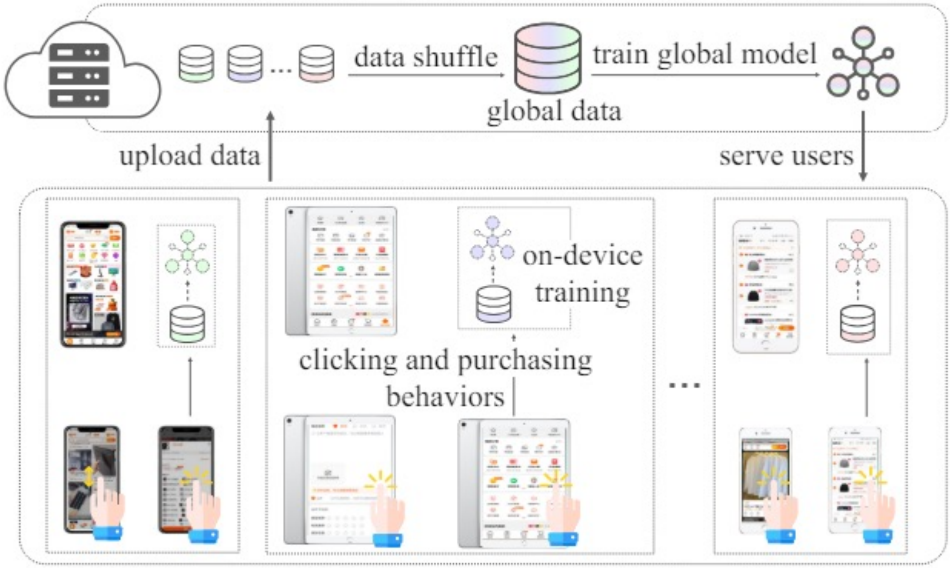



Fig. 6. Three types of on-device finetuning methods.

# Whole Model Finetuning

**MPDA** [Yan et al. 2022] enhances model personalization by leveraging large-scale cloud-coordinated domain adaptation, where external samples from the cloud are used to augment the user's local data for more effective on-device training.



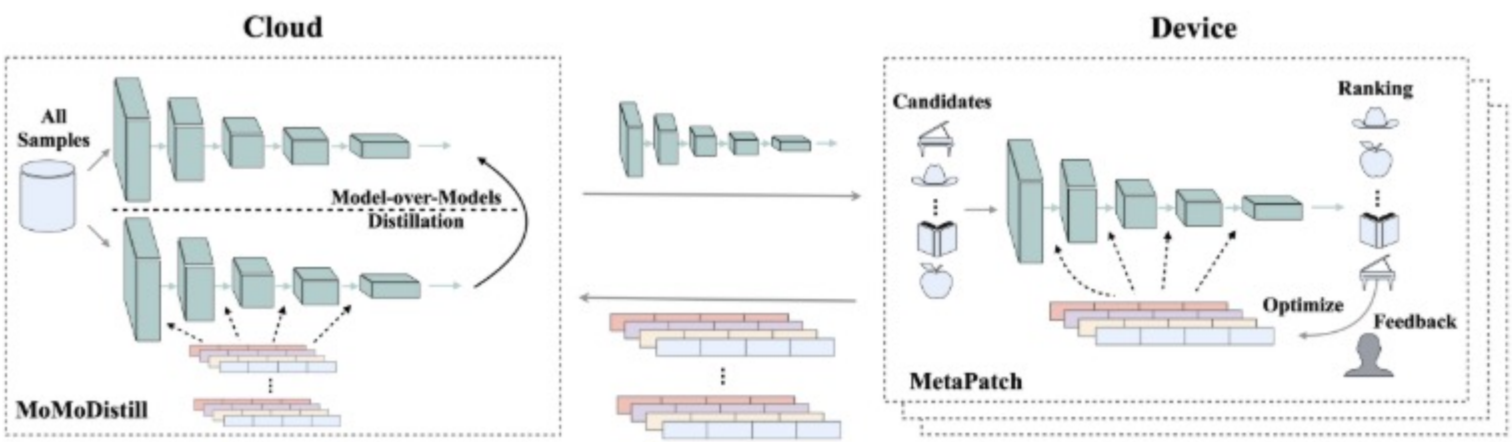

**Architecture of MPDA** [Yan et al. 2022]



Paper: Yan et al,. "On-Device Learning for Model Personalization with Large-Scale Cloud-Coordinated Domain Adaption". KDD. 2022.

# Patch Learning-based Finetuning

**DCCL** [Yao et al. 2021] introduces a novel approach for on-device personalization by adding parameter-efficient patches to a cloud model. Proposes a novel distillation technique that enhances the centralized cloud model by aggregating insights from numerous personalized device models.




**Architecture of DCCL** [Yao et al. 2021]



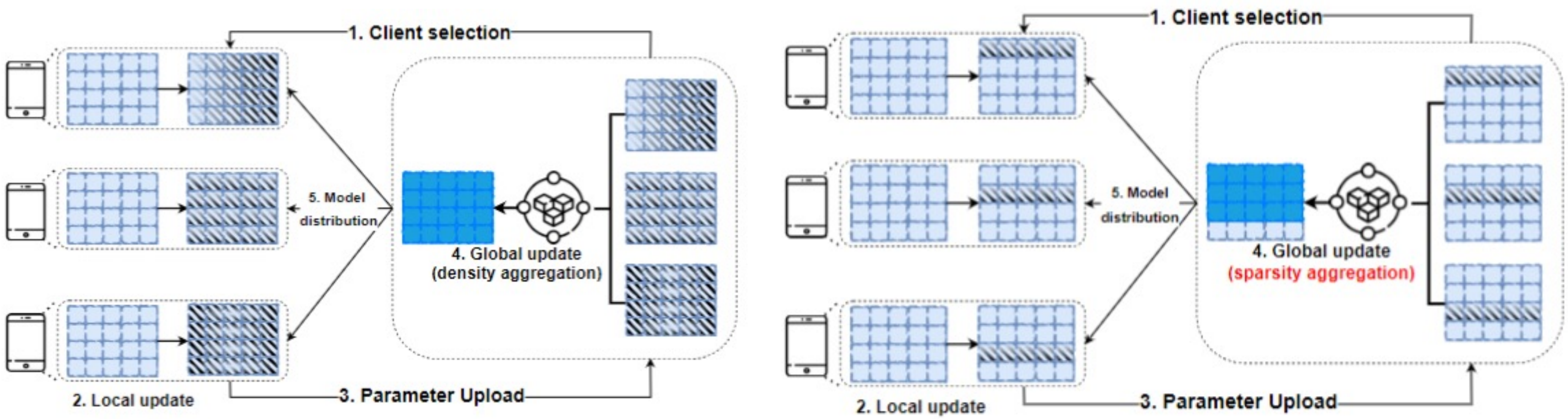
Paper: Yao et al., "Device-cloud collaborative learning for recommendation". KDD. 2021.

# User Privacy Risks

Behavioral Data Leakage

**FedGNN** [Wu et al. 2021] indicates that a central server with inquisitive intentions can easily identify rated items by analyzing non-zero gradients in recommendation with explicit feedback.

**FedMF** [Chai et al. 2020] indicates that by scrutinizing gradients sent by clients over two consecutive rounds, the central server can even infer the rating scores of items

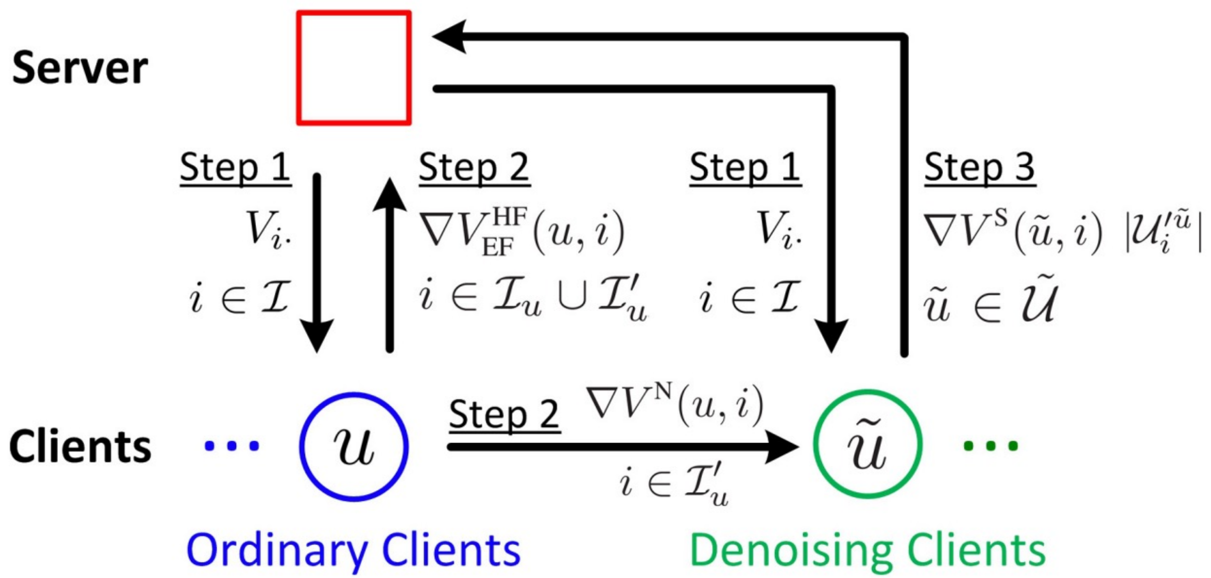



Papers: Wu et al., "Fedgnn: Federated graph neural network for privacy-preserving recommendation". 2021

Chai et al. "Secure federated matrix factorization." IEEE Intelligent Systems. 2020

# Data Obfuscation

Synthetic ratings [Lin et al. 2020]
Pseudo-labeling techniques [Liu et al. 2022]
**FedRec++** [Liang et al. 2021]**:** Denoising client




**Architecture of Fedrec++** [Liang et al. 2021]


Papers: Lin et al., "Fedrec: Federated recommendation with explicit feedback". IEEE Intelligent Systems, 2020
Liu et al. "Federated Social Recommendation with Graph Neural Network." ACM Trans. Intell. Syst. Technol. 2022
Liang et al. "Fedrec++: Lossless federated recommendation with explicit feedback" AAAI. 2021

# Model Obfuscation

Model obfuscation refers to the method that adds noise to the model gradients or model parameters.

- Local Differential Privacy [Wu et al. 2022]: Adds noise to data before it is sent to the server, such as
  - **Gradient Clipping:** Limits the magnitude of the gradients to a maximum value $\delta$
  - **Noise Addition:** Adds Laplacian noise to the clipped gradients with parameters

$$\mathbf{g}_i = clip(\mathbf{g}_i, \delta) + Laplace(0, \lambda)$$

Balancing between privacy protection and model accuracy.



Paper: Wu et al. "A federated graph neural network framework for privacy-preserving personalization". Nat Commun. 2022.

# Encryption-based Protection

- Encrypt parameters before uploading them to the central server
  - Homomorphic Encryption [Perifanis et al. 2023]
    - enables computational operations on encrypted data without the need for decryption
    - encryption algorithms typically increase the computational burden
  - Secure Multiparty Computation [Ying et al. 2020]
    - multiple clients/platforms to jointly compute a function over their inputs while keeping those inputs private
    - Computational and communication overhead can be significant

Papers: Perifanis et al., “FedPOIRec: Privacy-preserving federated poi recommendation with social influence”. Information Sciences 2023

Cui et al. “Exploiting data sparsity in secure cross-platform social recommendation”. NeurIPS 2021

# Poisoning Attacks and Countermeasures

Poisoning attacks involve deliberately inserting misleading or malicious data into a system to manipulate outcomes or degrade performance.

**Data Poisoning in ODRSs**: Attackers inject fake interactions or manipulate existing data to promote or demote products

**Model Poisoning in ODRSs**: Direct manipulation of the model's parameters by uploading poisoned updates under federated learning scenarios.


(a) Data poisoning attack for CloudRSs.

(b) Data poisoning attack for DeviceRSs.

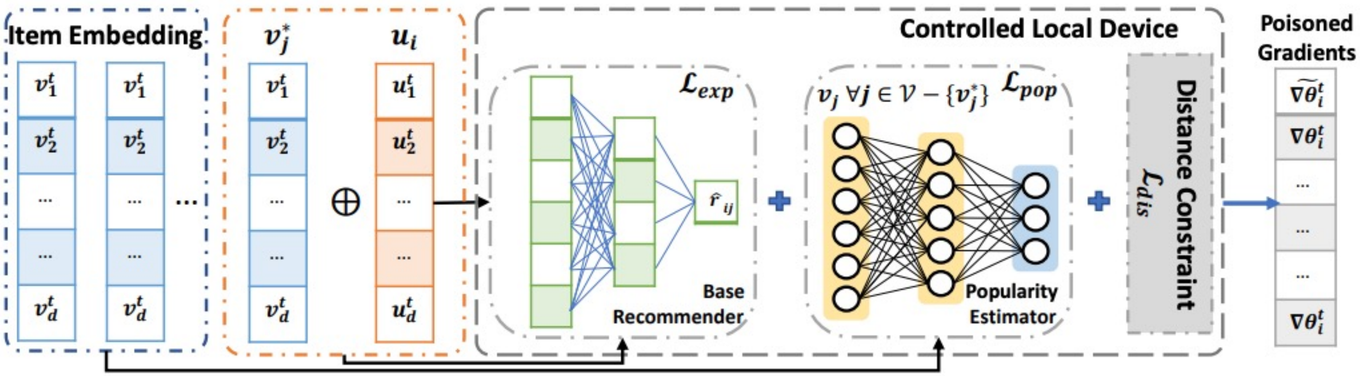
(c) Model poisoning attack for DeviceRSs.

Fig. 8. Overview of different attacks for CloudRSs and DeviceRSs.

# Model Poisoning Attacks

**PipAttack** [Zhang et al. 2023] achieves item promotion by aligning the embeddings of target items with popular items, requiring below conditions:

- The adversary can access the global model at any iteration
- The adversary can access and alter all malicious users' local models and their gradients.
- The adversary knows the whole item set (not interactions) which is commonly available on any e-commerce platform, as well as side information that reflects each item's popularity.




**Architecture of PipAttack** [Zhang et al. 2022]



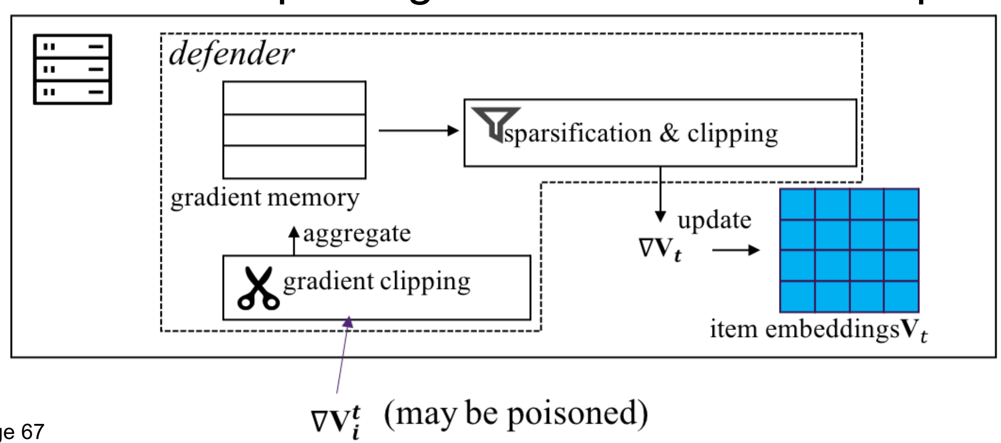
Paper: Zhang et al. "PipAttack: Poisoning Federated Recommender Systems for Manipulating Item Promotion". WSDM 2022

# Model Poisoning Attacks

**PSMU** [Yuan et al. 2023] constructs malicious users with random interactions, and promote target items by improving their prediction scores higher than the recommended items and alternative items.

E.g.,


**HiCS** [Yuan et al. 2023] utilizes two stages of gradient clipping and sparsification updating to dilute the effects of poisoned gradients.

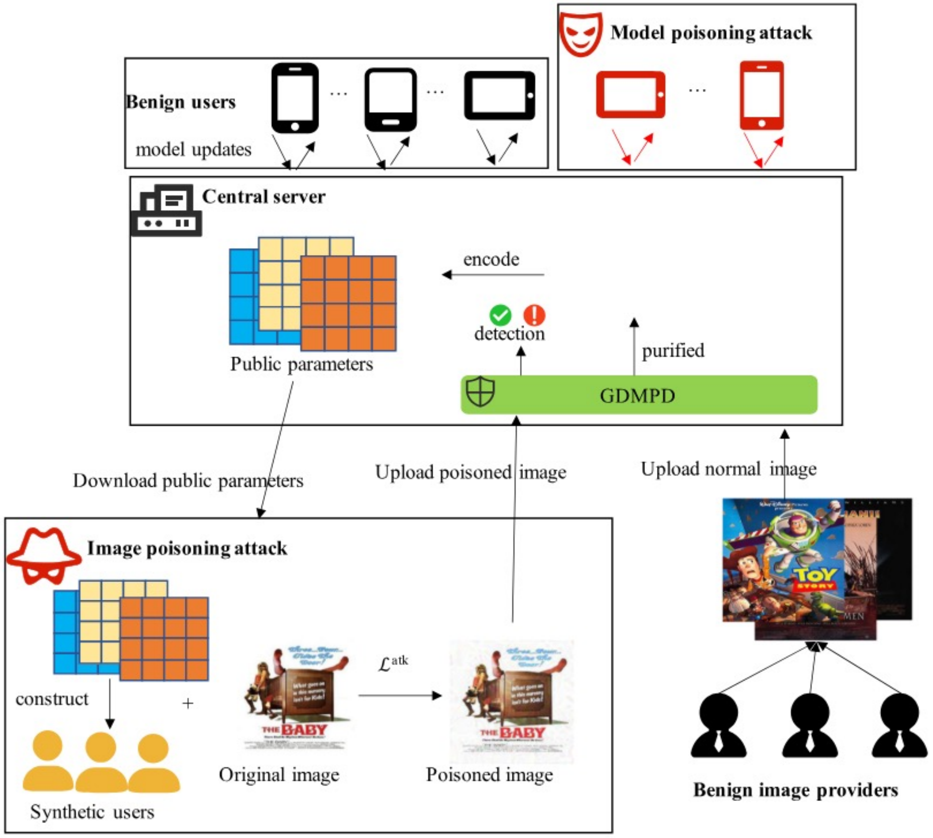





Paper: Yuan et al. “Manipulating federated recommender systems: Poisoning with synthetic users and its countermeasures”. SIGIR 2023.

# Hybrid Poisoning Attacks

**PSMU(V)** [Yuan et al. 2024] combines data poisoning attacks and model poisoning attacks in visually-aware FedRSs
They promote target items by contaminating both their visual signals (e.g., item posters) and item embeddings, highlighting the potential threats of incorporating third-party images in FedRSs.






Paper: Yuan et al. "Manipulating Visually-aware Federated Recommender Systems and Its Countermeasures". TOIS. 2023

The University of Queensland
Australia
Create Change

Chapter 1: Welcome and Introduction

Chapter 2: Definition and Taxonomy of ODRSs

Chapter 3: Deployment and Inference of ODRSs

Chapter 4: Training and Updating of ODRSs

Chapter 5: Security and Privacy of ODRSs

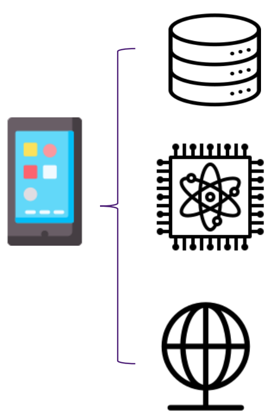
Chapter 6: Limitations and New Trends

Chapter 7: Open Discussions and More

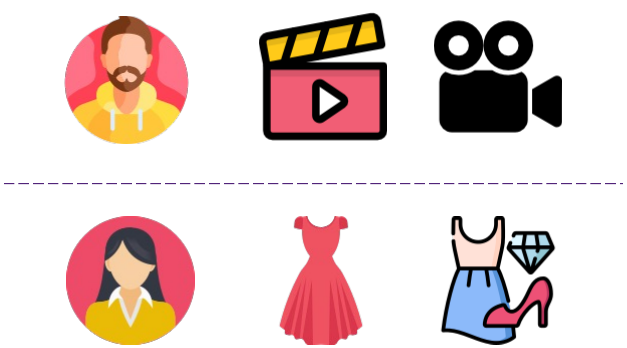
# Heterogeneity in ODRSs

Most ODRSs assume that each device/user is homogeneous, but this assumption is difficult to satisfy in real life due to the inherent heterogeneity among devices/users.

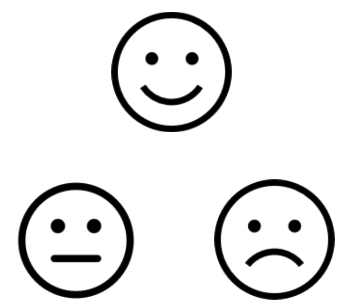
- System heterogeneity: storage, computation, and communication capabilities
- Data heterogeneity: data distribution, and user preferences
- Privacy heterogeneity: different privacy budget



**System heterogeneity**



**Data heterogeneity**

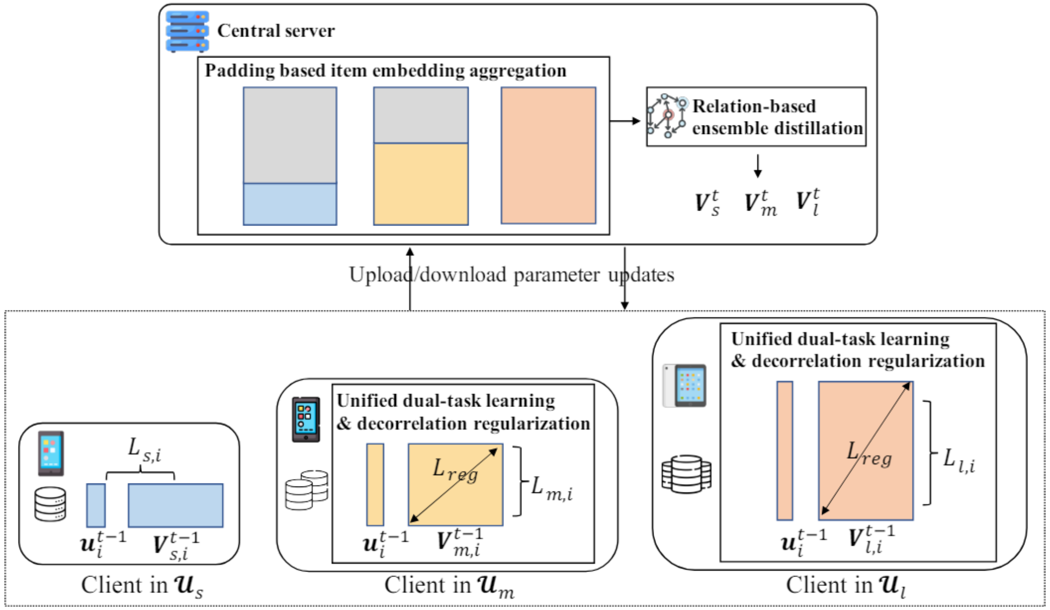


**Privacy heterogeneity**

# Heterogeneity in ODRSs

**HeteFedRec** [Yuan et al. 2024] is a novel framework for federated recommender systems that supports heterogeneous model sizes

It introduces a heterogeneous model aggregation strategy with dual-task learning and dimensional decorrelation regularization to enable efficient knowledge sharing among different-sized models.




**Architecture of HeteFedRec** [Yuan et al. 2024]

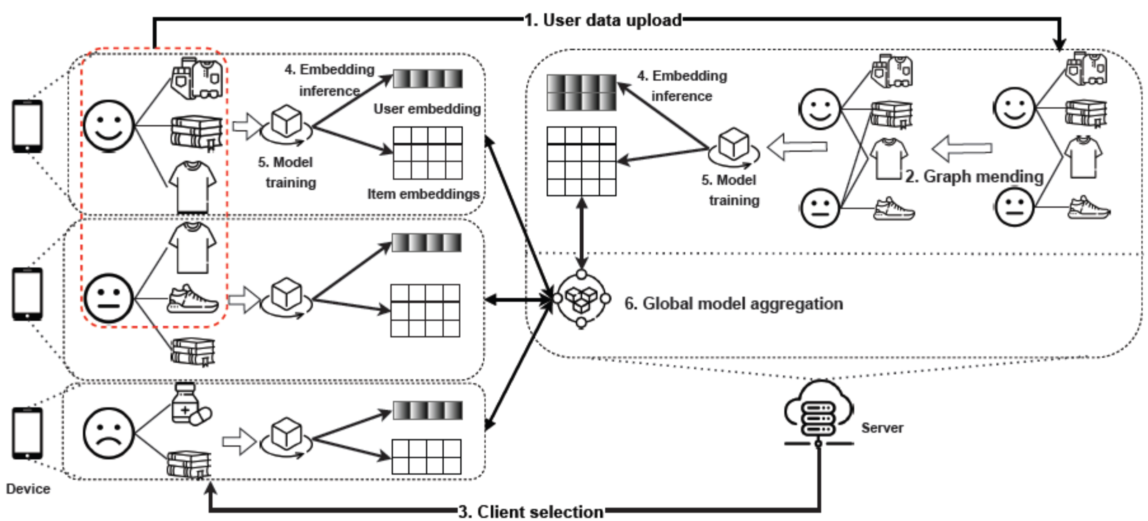


Paper: Yuan et al., "HeteFedRec: Federated Recommender Systems with Model Heterogeneity". ICDE 2024.

# Heterogeneity in ODRSs

**CDCGNNFed** [Qu et al. 2024] a novel framework for federated recommender systems that supports heterogeneous privacy budgets

- Users voluntarily choose to upload all, some, or no data to the server.
- Graph mending: The server employs a graph mending strategy to predict missing links.
- Train user-centric ego graphs locally, and high-order graphs based on user-shared data in the server in a collaborative manner via contrastive learning.




**Architecture of CDCGNNFed** [Qu et al. 2024]



Paper: Qu et al. "Towards Personalized Privacy: User-Governed Data Contribution for Federated Recommendation." WWW 2024.

# Evolving User Dynamics in ODRSs

Unlike traditional systems, ODRSs often experience changes in the user base, with new users joining and existing users leaving

- **Cold Start Problem in ODRSs**: it also needs to efficiently deploy models to newly added devices
- **Unlearning for ODRSs:** selectively forgetting data from users who are no longer active in the system [Yuan et al. 2023].


Image: Nicolò, et al. "Federated Unlearning: A Survey on Methods, Design Guidelines, and Evaluation Metrics." *arXiv* 2024

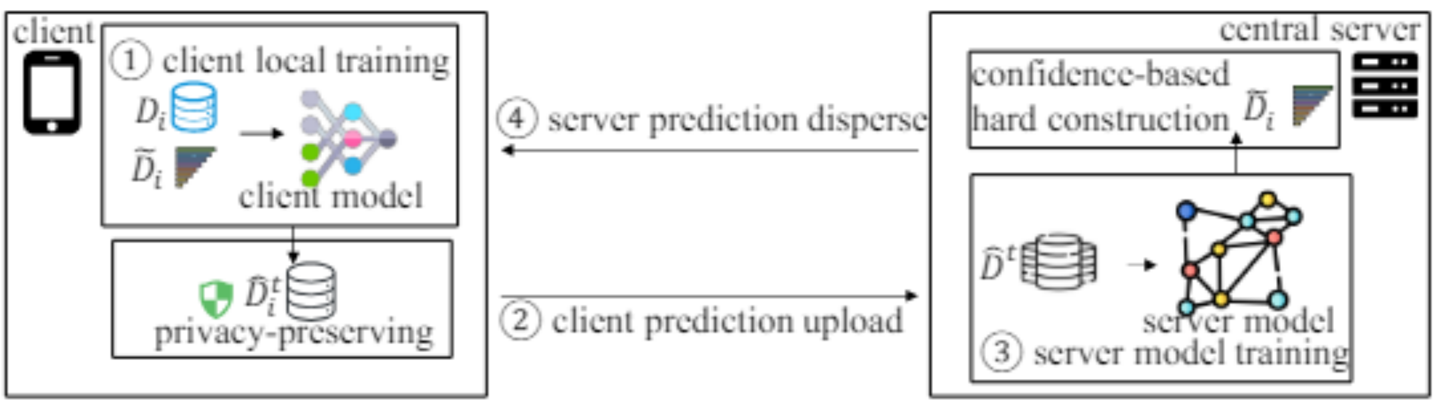
Yuan et al. “Federated unlearning for on-device recommendation. WSDM, 023 ”

# Model Copyright Protection in ODRSs

In ODRSs, recommender models are exposed to all users, increasing the risk of IP theft.

**PTF-FedRec** [Yuan et al. 2024] is a parameter transmission-free federated recommendation framework

- Achieves federated collaborative learning via sharing prediction scores over of a subset of items.
- Balance the protection of both clients' data privacy and the service provider's model privacy




**Architecture of PTF-FedRec** [Yuan et al. 2024]



Yuan et al., 2023. "Hide Your Model: A Parameter Transmission-free Federated Recommender System". ICDE 2024.

# Foundation Models in ODRSs

Current research primarily focuses on models that operate within cloud environments

- Cloud-based systems often suffer from delays in processing user requests, impacting user experience.
- The substantial computational requirements of these models make them difficult to deploy directly on user devices.

**Model Lightweighting**: Research into methods for reducing the size and complexity of foundation models to facilitate deployment on user devices.

**Privacy Considerations**: Local processing of data on devices could enhance user privacy by minimizing data transmission to the cloud.

THE UNIVERSITY OF QUEENSLAND AUSTRALIA | CREATE CHANGE

# Get More Information

## On-Device Recommender Systems: A Comprehensive Survey

HONGZHI YIN* and LIANG QU*, The University of Queensland, Australia
TONG CHEN, The University of Queensland, Australia
WEI YUAN, The University of Queensland, Australia
RUIQI ZHENG, The University of Queensland, Australia
JING LONG, The University of Queensland, Australia
XIN XIA, The University of Queensland, Australia
YUHUI SHI, Southern University of Science and Technology, China
CHENGQI ZHANG, The University of Technology Sydney, Australia

Recommender systems have been widely deployed in various real-world applications to help users identify content of interest from massive amounts of information. Traditional recommender systems work by collecting user-item interaction data in a cloud-based data center and training a centralized model to perform the recommendation service. However, such cloud-based recommender systems (CloudRSs) inevitably suffer from excessive resource consumption, response latency, as well as privacy and security risks concerning both data and models. Recently, driven by the advances in storage, communication, and computation capabilities of edge devices, there has been a shift of focus from CloudRSs to on-device recommender systems (DeviceRSs), which leverage the capabilities of edge devices to minimize centralized data storage requirements, reduce the response latency caused by communication overheads, and enhance user privacy and security by localizing data processing and model training. Despite the rapid rise of DeviceRSs, there is a clear absence of timely literature reviews that systematically introduce, categorize and contrast these methods. To bridge this gap, we aim to provide a comprehensive survey of DeviceRSs, covering three main aspects: (1) the deployment and inference of DeviceRSs, exploring how large recommendation models can be compressed and utilized within resource-constrained on-device environments; (2) the training and update of DeviceRSs, discussing how local data can be leveraged for model optimization on the device side; (3) the security and privacy of DeviceRSs, unveiling their potential vulnerability to malicious attacks and defensive strategies to safeguard these systems. Furthermore, we provide a fine-grained and systematic taxonomy of the methods involved in each aspect, followed by a discussion regarding challenges and future research directions. This is the first comprehensive survey on DeviceRSs that covers a spectrum of tasks to fit various needs. We believe this survey will help readers effectively grasp the current research status in this field, equip them with relevant technical foundations, and stimulate new research ideas for developing DeviceRSs.

https://arxiv.org/abs/2401.11441

# Special Issue CFP

SciEngine

Journals & Books

Home | Archive | Author Center | Editorial Board | About | News | Moop | Advanced Search

## Call for Papers: Cloud-Edge Collaboration for On-Device Recommendation

2024-02-02 PageView: 440

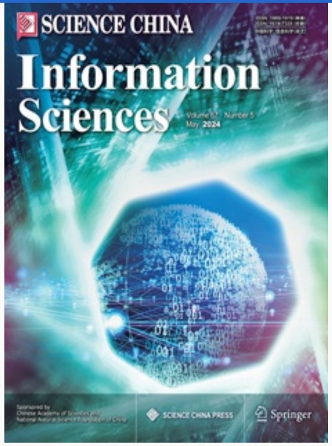
### Important Dates

- Open for submissions: 8 April 2024
- Submissions due: 31 May 2024
- Preliminary notification: 1 August 2024
- Acceptance notification: 1 November 2024
- Final version due: 15 January 2025
- Planned publication: Late 2025

### Topic Summary

Given the sheer volume of contemporary e-commerce applications, recommender systems (RSs) have gained significant attention in both academia and industry. However, traditional cloud-based RSs face inevitable challenges such as resource-intensive computation, reliance on network access, and privacy breaches. In response, a new paradigm called on-device recommender systems (ODRSs) has emerged in recent years, and has found applications in various industries like Taobao, Google, and Kuaishou. Usually subsumed under a cloud-edge collaboration paradigm, ODRSs unleash the computational capacity of user devices with lightweight recommendation models tailored for resource-constrained environments, enabling real-time inference with users' local data. With the increasing prevalence of cloud-edge collaboration for ODRSs, this special issue aims to stimulate and collate cutting-edge research on this new recommendation paradigm, where the topics of interest include, but are not limited to:

- Model compression techniques for efficient storage of recommender systems
- Lightweight recommendation models for memory-efficient and fast on-device inference
- Federated or fully decentralized recommender systems with edge devices
- Incremental on-device learning for recommender systems




| | |
|---|---|
| Impact Factor | 8.8 |
| Five-year Impact Factor | 5.6 |
| CSCD Impact Factor | 0.9222 |
| CiteScore | 10.1 |

Editor-in-Chief

Mei Hong

Science China Information Science
https://www.sciengine.com/SCIS/newsDetails?slug=newsDetails&abbreviated=scp&specialId=73c5ff8ad2924f86bb41d1df936da116

# Thanks
# Q&A

# References


(OAIC 2023) https://www.oaic.gov.au/__data/assets/pdf_file/0021/156531/Notifiable-data-breaches-report-July-to-December-2023.pdf

(ACCC 2023) https://www.accc.gov.au/by-industry/telecommunications-and-internet/mobile-services-regulation/mobile-infrastructure-report/mobile-infrastructure-report-2023

(Jones 2018) Jones, Nicola. "How to stop data centres from gobbling up the world's electricity." *Nature* 561, no. 7722 (2018): 163-166.

(Noia et al. 2022) Noia, Tommaso Di, Nava Tintarev, Panagiota Fatourou, and Markus Schedl. "Recommender systems under European AI regulations." Communications of the ACM 65, no. 4 (2022): 69-73.

(Ge et al. 2022) Ge, Yingqiang, Shuchang Liu, Zuohui Fu, Juntao Tan, Zelong Li, Shuyuan Xu, Yunqi Li, Yikun Xian, and Yongfeng Zhang. "A survey on trustworthy recommender systems." *ACM Transactions on Recommender Systems* (2022).

(Chen et al. 2021) Chen, Tong, Hongzhi Yin, Yujia Zheng, Zi Huang, Yang Wang, and Meng Wang. "Learning elastic embeddings for customizing on-device recommenders." In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 138-147. 2021.

(Wang et al. 2020) Wang, Qinyong, Hongzhi Yin, Tong Chen, Zi Huang, Hao Wang, Yanchang Zhao, and Nguyen Quoc Viet Hung. "Next point-of-interest recommendation on resource-constrained mobile devices." In *Proceedings of the Web conference 2020*, pp. 906-916. 2020.

(Yin et al. 2024) Yin, Hongzhi, Liang Qu, Tong Chen, Wei Yuan, Ruiqi Zheng, Jing Long, Xin Xia, Yuhui Shi, and Chengqi Zhang. "On-Device Recommender Systems: A Comprehensive Survey." *arXiv preprint arXiv:2401.11441* (2024).

(Gong et al. 2022) Gong, Xudong, Qinlin Feng, Yuan Zhang, Jiangling Qin, Weijie Ding, Biao Li, Peng Jiang, and Kun Gai. "Real-time short video recommendation on mobile devices." In *Proceedings of the 31st ACM international conference on information & knowledge management*, pp. 3103-3112. 2022.

(TFL 2024) TensorFlow Lite Recommendation. https://www.tensorflow.org/lite/examples/recommendation/overview

(Gong et al. 2020) Gong, Yu, Ziwen Jiang, Yufei Feng, Binbin Hu, Kaiqi Zhao, Qingwen Liu, and Wenwu Ou. "EdgeRec: recommender system on edge in Mobile Taobao." In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 2477-2484. 2020.

# References

(Minto et al. 2021) Using Federated Learning to Improve Brave's On-Device Recommendations While Protecting Your Privacy. https://brave.com/blog/federated-learning/

(Koren et al. 2009) Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." *Computer*42, no. 8 (2009): 30-37.

(Rendle 2011) Rendle, Steffen. "Factorization machines." In *2010 ICDM*, pp. 995-1000. IEEE, 2010.

(He et al. 2017a) He, Xiangnan, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. "Neural collaborative filtering." In *Proceedings of the 26th international conference on world wide web*, pp. 173-182. 2017.

(He et al. 2017b) He, Xiangnan, and Tat-Seng Chua. "Neural factorization machines for sparse predictive analytics." In *Proceedings of the 40th International ACM SIGIR*, pp. 355-364. 2017.

(He et al. 2020) He, Xiangnan, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. "Lightgcn: Simplifying and powering graph convolution network for recommendation." In *Proceedings of the 43rd International ACM SIGIR*, pp. 639-648. 2020.

(Joglekar et al. 2020) Joglekar, Manas R., Cong Li, Mei Chen, Taibai Xu, Xiaoming Wang, Jay K. Adams, Pranav Khaitan, Jiahui Liu, and Quoc V. Le. "Neural input search for large scale recommendation models." In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2387-2397. 2020.

(Zhang et al. 2016) Zhang, Hanwang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. "Discrete collaborative filtering." In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 325-334. 2016.

(Liang et al. 2023) Liang, Xurong, Tong Chen, Quoc Viet Hung Nguyen, Jianxin Li, and Hongzhi Yin. "Learning compact compositional embeddings via regularized pruning for recommendation." In *2023 IEEE ICDM*, pp. 378-387. IEEE, 2023.

(Liang et al. 2024) Liang, Xurong, Tong Chen, Lizhen Cui, Yang Wang, Meng Wang, and Hongzhi Yin. "Lightweight Embeddings for Graph Collaborative Filtering." *SIGIR 2024.*

(Wang et al. 2020) Wang, Qinyong, Hongzhi Yin, Tong Chen, Zi Huang, Hao Wang, Yanchang Zhao, and Nguyen Quoc Viet Hung. "Next point-of-interest recommendation on resource-constrained mobile devices." In *Proceedings of the Web conference 2020*, pp. 906-916. 2020.

(Wang et al. 2018) J. Wang et al., Billion-scale commodity embedding for e-commerce recommendation in alibaba. KDD, 2018.

# References

(Eksombatchai et al. 2018) Eksombatchai, Chantat, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. "Pixie: A system for recommending 3+ billion items to 200+ million users in real-time." In *Proceedings of the 2018 world wide web conference*, pp. 1775-1784. 2018.
(Zhou et al. 2012) Zhou, Ke, and Hongyuan Zha. "Learning binary codes for collaborative filtering." In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 498-506. 2012.
(Tan et al. 2020) Tan, Qiaoyu, Ninghao Liu, Xing Zhao, Hongxia Yang, Jingren Zhou, and Xia Hu. "Learning to hash with graph neural networks for recommender systems." In *Proceedings of The Web Conference 2020*, pp. 1988-1998. 2020.
(Yoshua et al. 2013) Bengio, Yoshua, Nicholas Léonard, and Aaron Courville. "Estimating or propagating gradients through stochastic neurons for conditional computation." *arXiv preprint arXiv:1308.3432* (2013).
(Sedaghati et al. 2015) Sedaghati, Naser, Te Mu, Louis-Noel Pouchet, Srinivasan Parthasarathy, and P. Sadayappan. "Automatic selection of sparse matrix representation on GPUs." In *Proceedings of the 29th ACM on International Conference on Supercomputing*, pp. 99-108. 2015.
(Virtanen et al. 2020) Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski et al. "SciPy 1.0: fundamental algorithms for scientific computing in Python." *Nature methods* 17, no. 3 (2020): 261-272.
(Liu et al. 2021) Liu, Siyi, Chen Gao, Yihong Chen, Depeng Jin, and Yong Li. "Learnable embedding sizes for recommender systems." *ICLR 2021.*
(Lyu et al. 2022) Lyu, Fuyuan, Xing Tang, Hong Zhu, Huifeng Guo, Yingxue Zhang, Ruiming Tang, and Xue Liu. "Optembed: Learning optimal embedding table for click-through rate prediction." In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 1399-1409. 2022.
(Qu et al. 2022) Qu, Liang, Yonghong Ye, Ningzhi Tang, Lixin Zhang, Yuhui Shi, and Hongzhi Yin. "Single-shot embedding dimension search in recommender system." In *Proceedings of the 45th International ACM SIGIR conference on research and development in Information Retrieval*, pp. 513-522. 2022.
(Zhao et al. 2021) Zhao, Xiangyu, Haochen Liu, Wenqi Fan, Hui Liu, Jiliang Tang, Chong Wang, Ming Chen, Xudong Zheng, Xiaobing Liu, and Xiwang Yang. "Autoemb: Automated embedding dimensionality search in streaming recommendations." In *2021 IEEE International Conference on Data Mining (ICDM)*, pp. 896-905. IEEE, 2021.

# References


(Liu et al. 2018) Liu, Hanxiao, Karen Simonyan, and Yiming Yang. "DARTS: Differentiable Architecture Search." In *International Conference on Learning Representations*. 2018.

(Yan et al. 2021) Yan, Bencheng, Pengjie Wang, Kai Zhang, Wei Lin, Kuang-Chih Lee, Jian Xu, and Bo Zheng. "Learning effective and efficient embedding via an adaptively-masked twins-based layer." In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 3568-3572. 2021.

(Real et al. 2019) Real, Esteban, Alok Aggarwal, Yanping Huang, and Quoc V. Le. "Regularized evolution for image classifier architecture search." In *Proceedings of the aaai conference on artificial intelligence*, vol. 33, no. 01, pp. 4780-4789. 2019.

(Zheng et al. 2024) Zheng, Ruiqi, Liang Qu, Tong Chen, Kai Zheng, Yuhui Shi, and Hongzhi Yin. "Personalized Elastic Embedding Learning for On-Device Recommendation." *IEEE Transactions on Knowledge and Data Engineering* (2024).

(Qu et al. 2023) Qu, Yunke, Tong Chen, Xiangyu Zhao, Lizhen Cui, Kai Zheng, and Hongzhi Yin. "Continuous input embedding size search for recommender systems." In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 708-717. 2023.

(Fujimoto et al. 2018) Fujimoto, Scott, Herke Hoof, and David Meger. "Addressing function approximation error in actor-critic methods." In *International conference on machine learning*, pp. 1587-1596. PMLR, 2018.

(Shi et al. 2020) Shi, Hao-Jun Michael, Dheevatsa Mudigere, Maxim Naumov, and Jiyan Yang. "Compositional embeddings using complementary partitions for memory-efficient recommendation systems." In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 165-175. 2020.

(Xia et al. 2022) Xia, Xin, Hongzhi Yin, Junliang Yu, Qinyong Wang, Guandong Xu, and Quoc Viet Hung Nguyen. "On-device next-item recommendation with self-supervised knowledge distillation." In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 546-555. 2022.

(Hinton et al. 2012) Hinton, Geoffrey E., Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. "Improving neural networks by preventing co-adaptation of feature detectors." *arXiv preprint arXiv:1207.0580* (2012).

# References

(Yao et al. 2021) Yao, Jiangchao, Feng Wang, Kunyang Jia, Bo Han, Jingren Zhou, and Hongxia Yang. "Device-cloud collaborative learning for recommendation." In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 3865-3874. 2021.

(Rendle et al. 2009) Rendle, Steffen, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. "BPR: Bayesian personalized ranking from implicit feedback." In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 452-461. 2009.

(Zhou et al. 2018) Zhou, Guorui, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. "Deep interest network for click-through rate prediction." In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1059-1068. 2018.

(Chen et al. 2020) Chen, Tong, Hongzhi Yin, Quoc Viet Hung Nguyen, Wen-Chih Peng, Xue Li, and Xiaofang Zhou. "Sequence-aware factorization machines for temporal predictive analytics." In *2020 IEEE 36th international conference on data engineering (ICDE)*, pp. 1405-1416. IEEE, 2020.

(Xia et al. 2023) Xia, Xin, Junliang Yu, Qinyong Wang, Chaoqun Yang, Nguyen Quoc Viet Hung, and Hongzhi Yin. "Efficient on-device session-based recommendation." *ACM Transactions on Information Systems* 41, no. 4 (2023): 1-24.