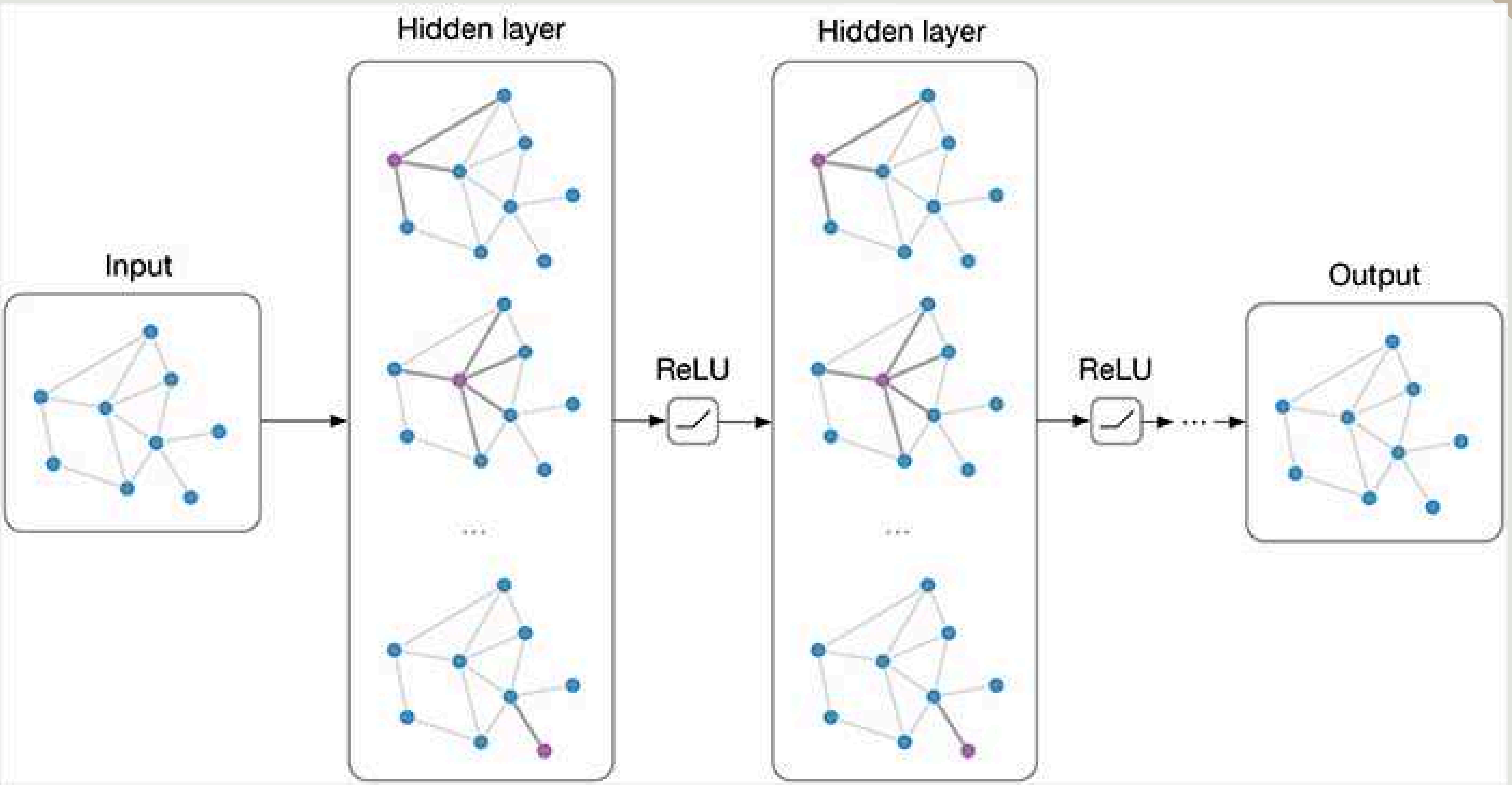
GRAPH-LESS COLLABORATIVE FILTERING

Member:
21127327 - Nguyễn Trần Trung Kiên
21127329 - Châu Tấn Kiệt
21127170 - Nguyễn Thế Thiện

Instructors:
Thầy Bùi Duy Đăng
Cô Nguyễn Ngọc Thảo

GNN

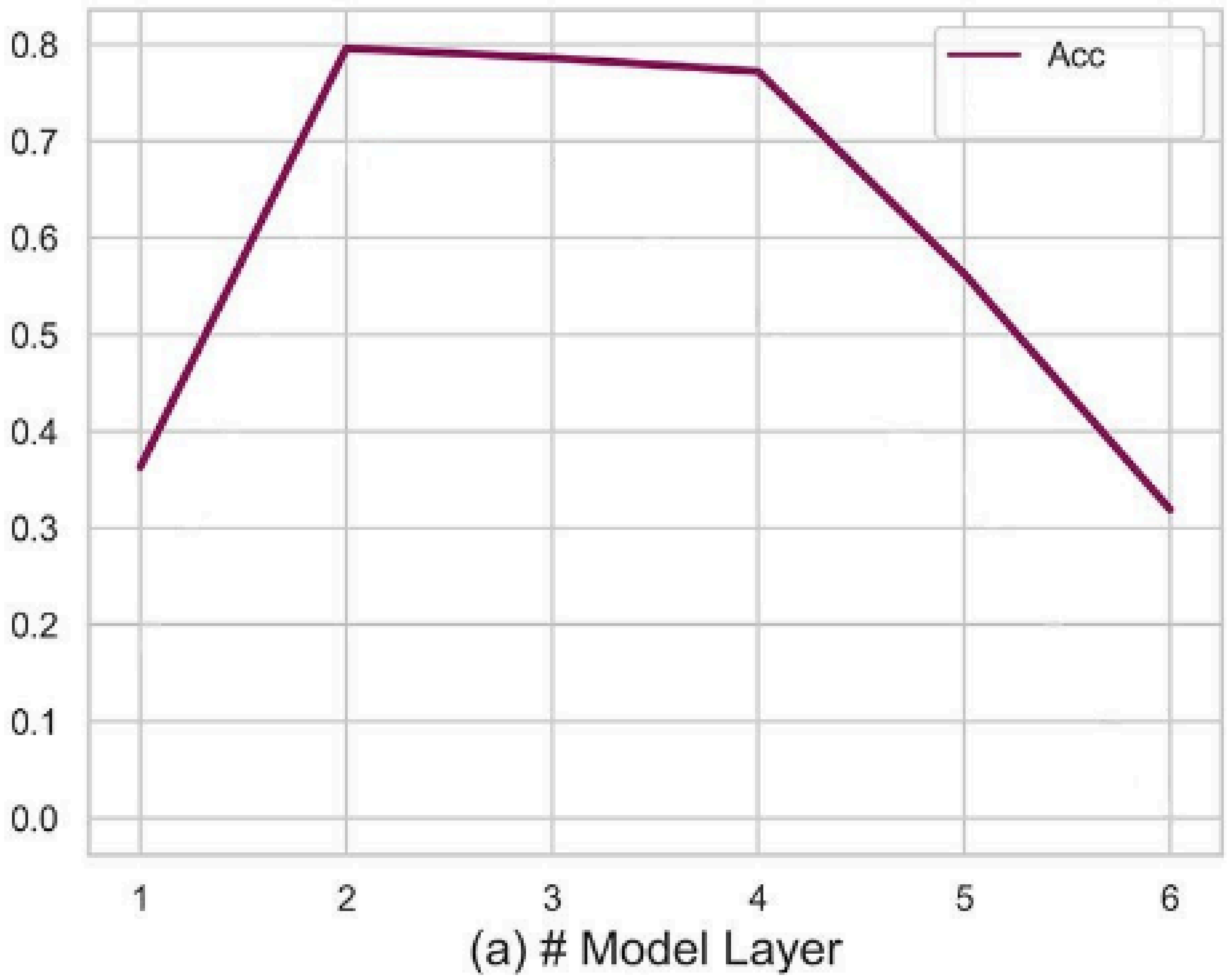
Franco Scarselli et. al. in 2008



Problem: over-smoothing

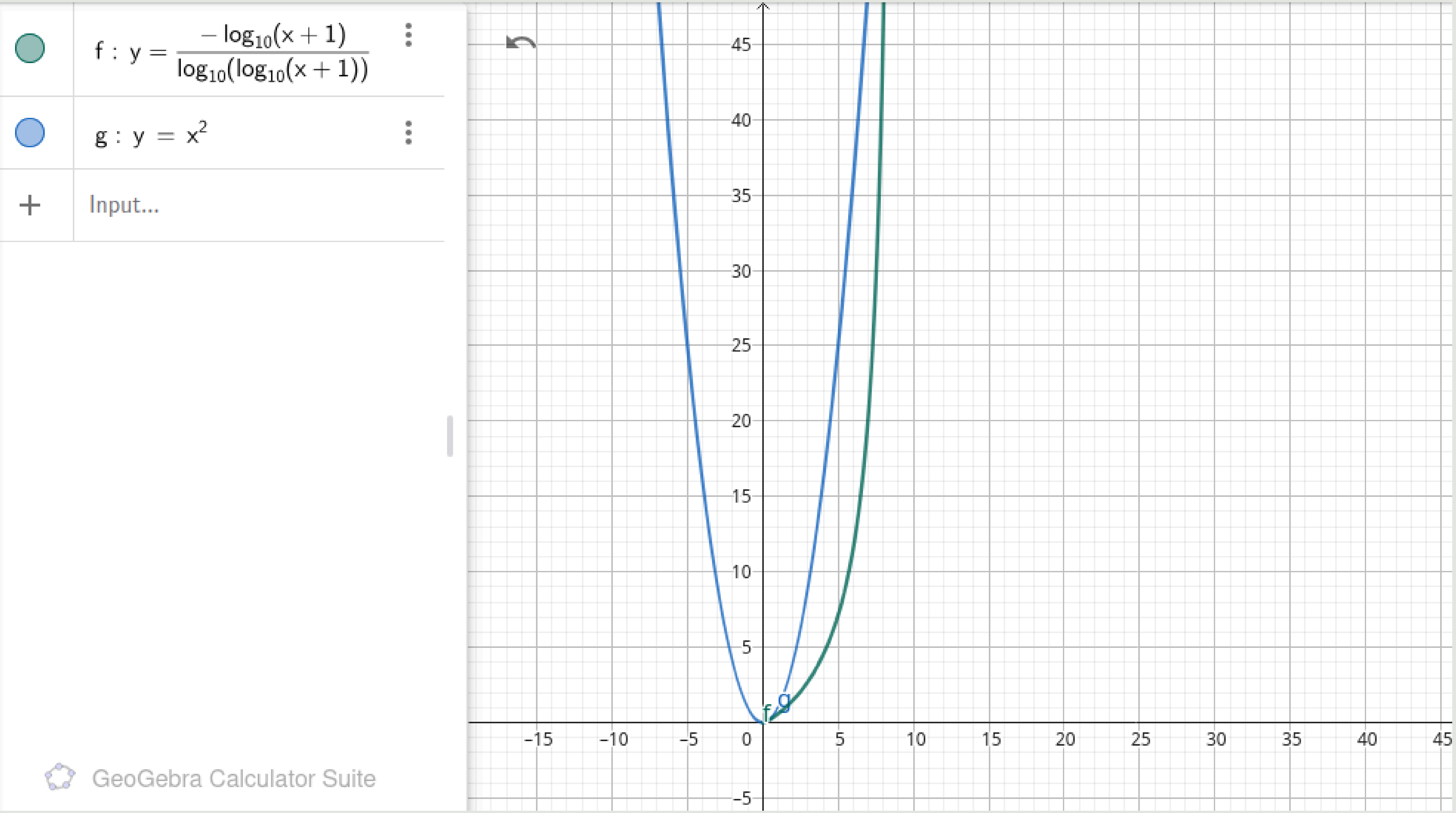
$$O\left(\frac{\log N}{\log\log N}\right)$$

Xinyi Wu, et. al. (2023).



The node classification accuracy (Acc) of GCNs on the CORA dataset.

Problem: over-smoothing



Background and Problem

Graph Neural Networks (GNNs) are gaining popularity in recommender systems due to their ability to model user-item interactions.

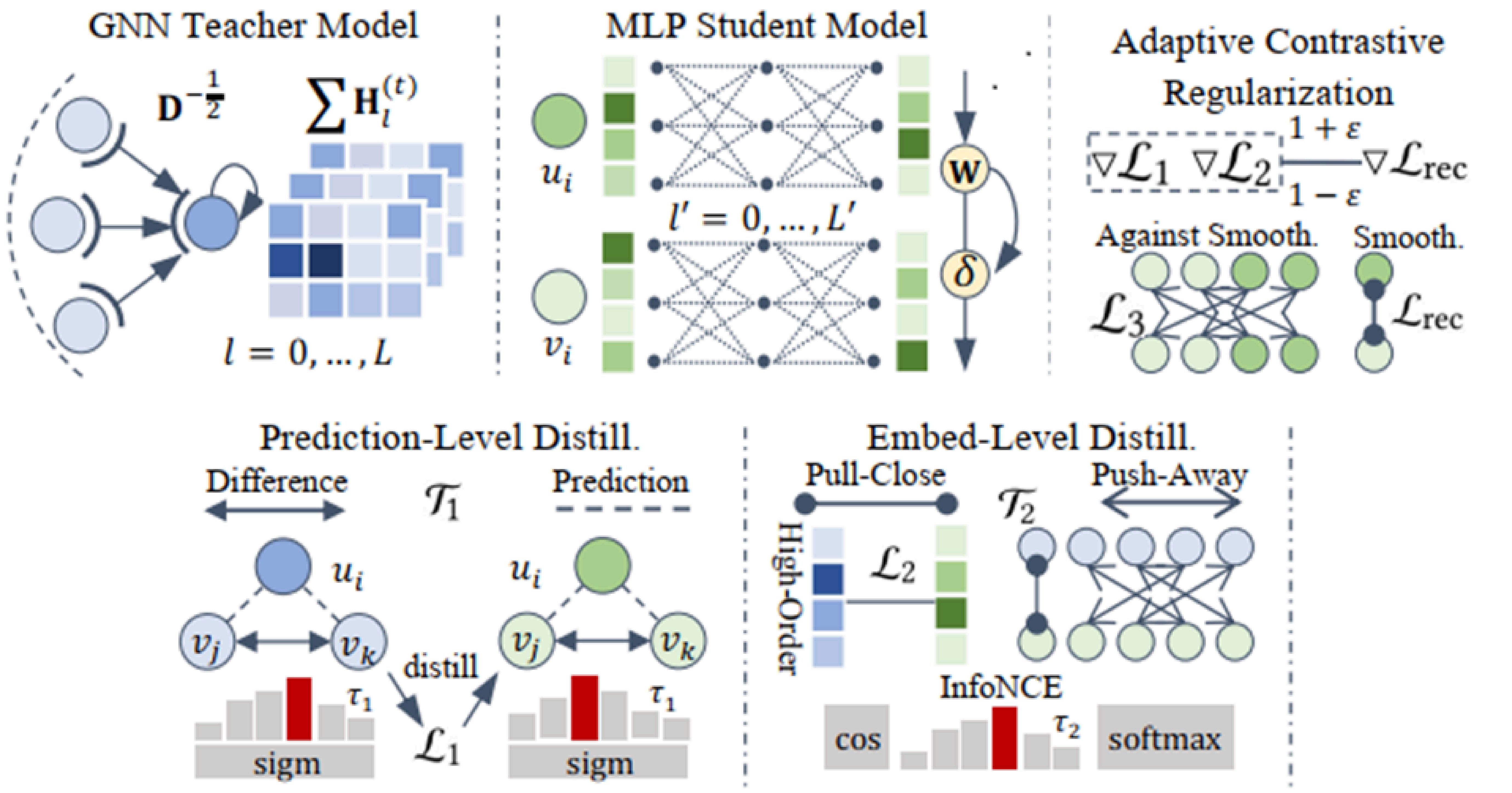
However, GNNs have several serious problems:

- **Over-smoothing phenomenon**
- **Noise propagation**
- **High computational cost (especially on large graphs)**

Background and Problem

Is it possible to build a recommender model without graph propagation (graph-less), but still learn complex features like GNN model?

SimRec Model



SimRec Model

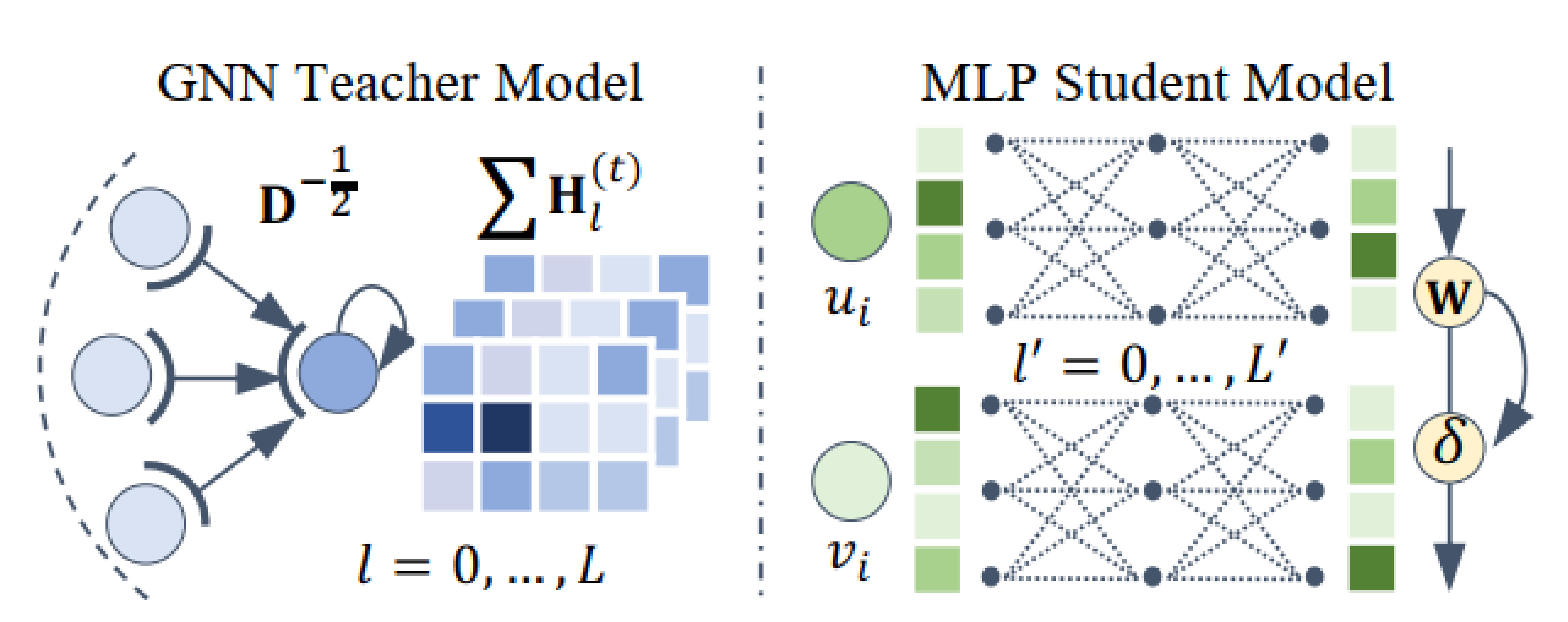
The author proposes the SimRec model – a recommendation model that does not use GNN in inference:
Based on a lightweight, fast MLP architecture.
Combining 2 main strategies:

- **Knowledge Distillation (2 levels):**
  - **Prediction-level (L1): learns from the ranking results from the GNN teacher.**
  - **Embedding-level (L2): learns from the representation from the GNN teacher.**
- **Adaptive Contrastive Regularization (L3): Helps to distinguish embeddings better, avoiding over-smoothing.**

Methodology

- Contrastive Knowledge Distillation
- Adaptive Contrastive Regularization
- Student Loss Function
- Further Discussion of SimRec

Contrastive Knowledge Distillation



Contrastive Knowledge Distillation

SimRec consists of two main models:

- **Teacher Model (GNN-based):**
  - **Uses GNN (e.g. LightGCN) to learn embeddings from user-item graphs.**
  - **Capable of modeling high-order information through graph propagation.**
  - **Pretrained.**
- **Student Model (MLP-based):**
  - **Is a lightweight MLP network, without graph propagation.**
  - **Relearns teacher behavior through distillation and regularization.**

Contrastive Knowledge Distillation

Prediction-level Distillation (L1)

Goal: Transfer knowledge from the teacher's predictions (soft signals) to the student.

- The student learns to create a ranking order similar to the teacher.
- This type of distillation helps the student model understand how the teacher evaluates user–item pairs.

Embedding-level Distillation (L2)

Goal: Make the embedding learned by the student closer to the teacher's embedding.

- Performed on both user and item embeddings.
- This ensures that the learned representation does not deviate too far in semantics from the teacher GNN.

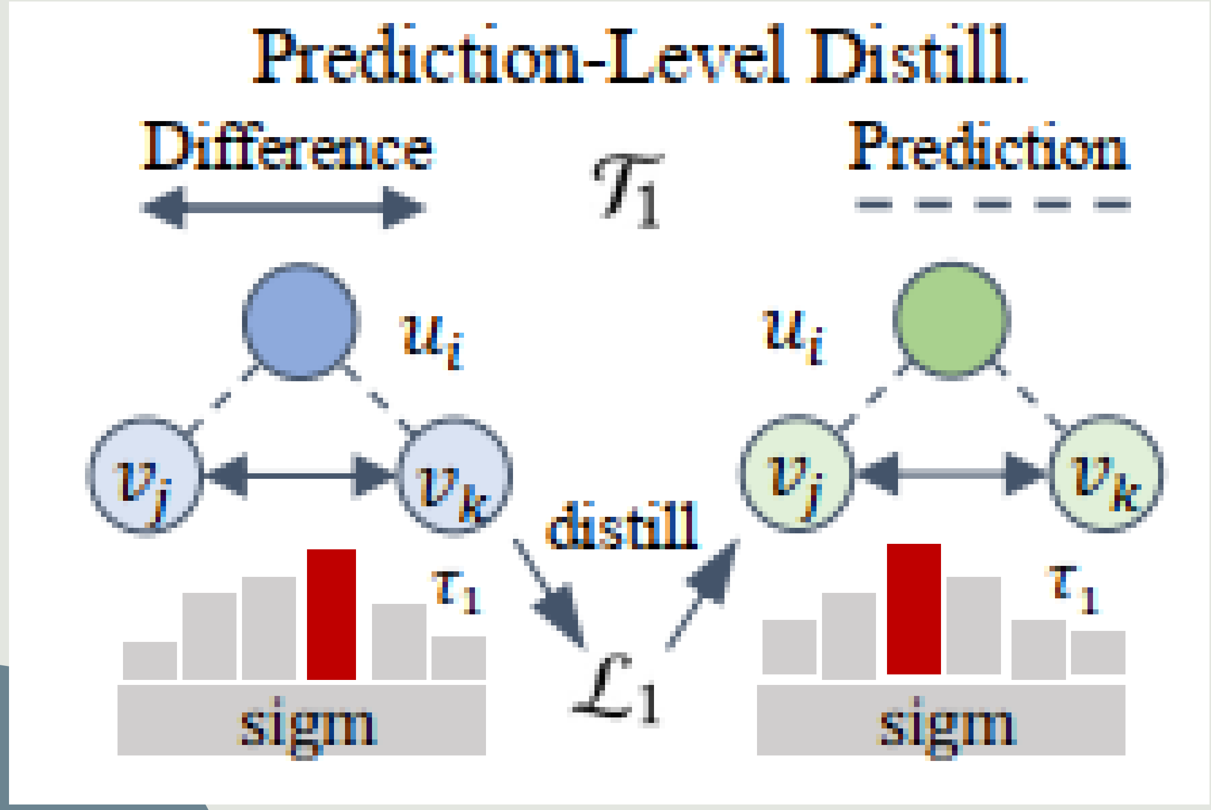
Contrastive Regularization (L3)

Goal: Increase the discriminativeness between embedding nodes.

- Avoid over-smoothing – when embeddings become too similar.
- Irrelevant nodes will be "pushed away" in the embedding space, increasing the discriminability.

Prediction-level Distillation (L1)

- We calculate the score difference between the two items for each user from the set of triples (user, item$^+$, item$^-$) sampled from the entire dataset.
- Teacher (GNN) and Student (MLP) both calculate this difference, which is then normalized using a sigmoid function with a temperature coefficient (τ_1) to "smooth" the value.
- Using a loss function to force the teacher and student distributions to be as close as possible.

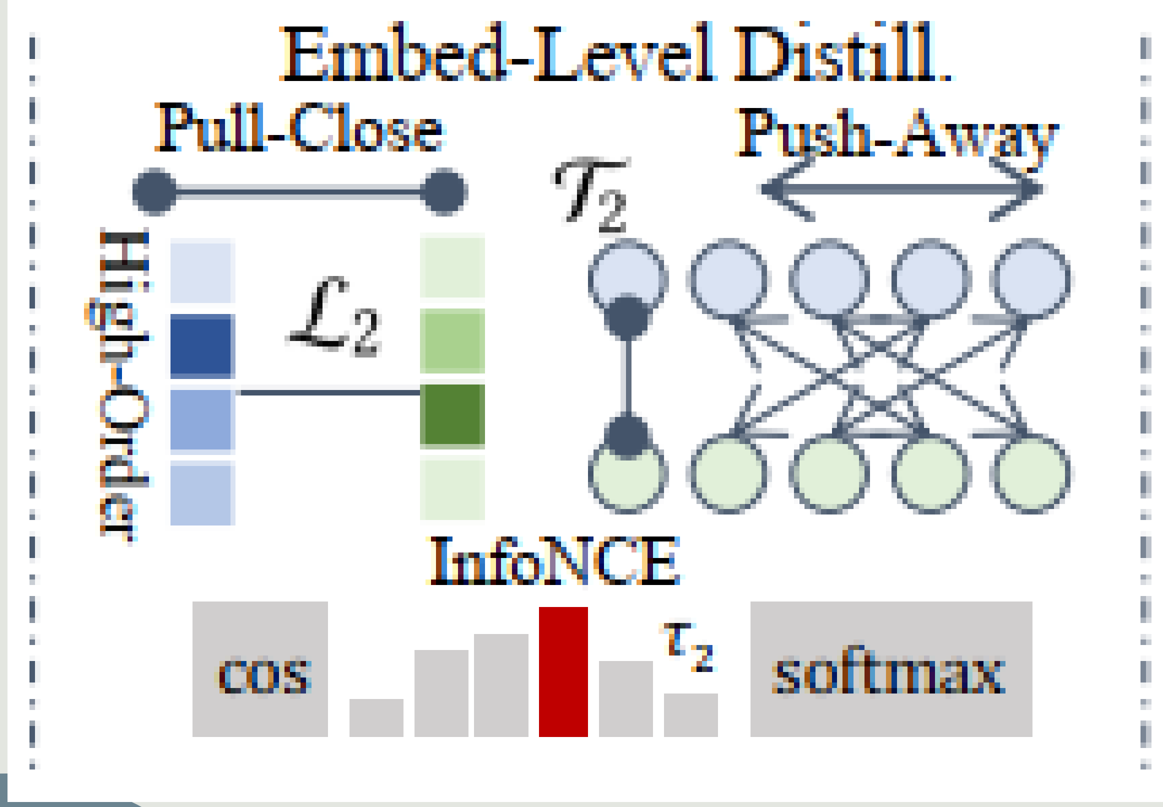


$$\mathcal{L}_1 = \sum_{(u_i, v_j, v_k) \in \mathcal{T}_1} -\left(\bar{z}^{(t)}_{i,j,k} \cdot \log \bar{z}^{(s)}_{i,j,k} + (1 - \bar{z}^{(t)}_{i,j,k}) \cdot \log(1 - \bar{z}^{(s)}_{i,j,k})\right)$$

$$\bar{z}^{(t)}_{i,j,k} = \mathrm{sigm}(z^{(t)}_{i,j,k}/\tau_1), \quad \bar{z}^{(s)}_{i,j,k} = \mathrm{sigm}(z^{(s)}_{i,j,k}/\tau_1) \qquad (7)$$

Embedding-level Distillation (L2)

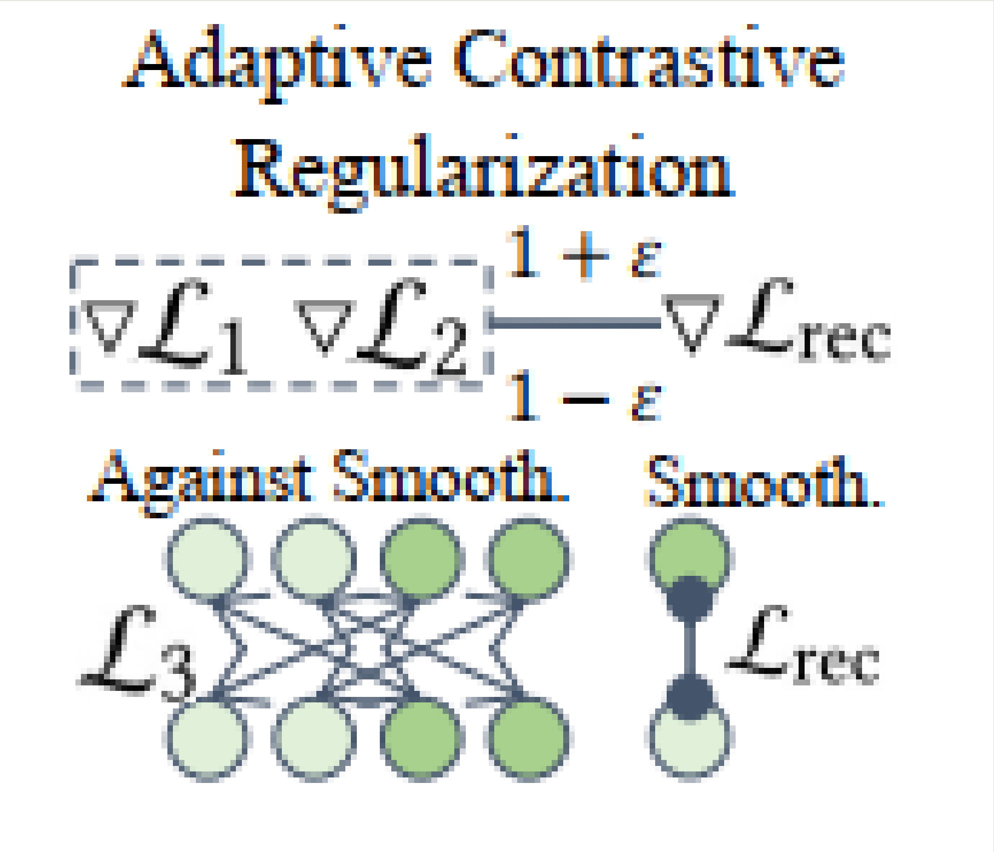
- Apply similarity measures such as cosine similarity combined with contrastive loss (InfoNCE) – compare the embedding of a user (or item) generated by the MLP with the corresponding "synthetic" embedding from the teacher.
- Use the temperature coefficient τ_2 in the softmax function to adjust the "sharpness" of the distribution, thereby pressuring the student to stick to the hidden structure that the teacher has learned.



$$\mathcal{L}_2 = \sum_{u_i \in \mathcal{T}_2} -\log \frac{\exp\left(\cos(\mathbf{h}_i^{(s)}, \sum_{l=2}^{L} \mathbf{h}_{i,l}^{(t)})/\tau_2\right)}{\sum_{u_{i'} \in \mathcal{U}} \exp\left(\cos(\mathbf{h}_{i'}^{(s)}, \sum_{l=2}^{L} \mathbf{h}_{i,l}^{(t)})/\tau_2\right)} + \sum_{v_j \in \mathcal{T}_2} -\log \frac{\exp\left(\cos(\mathbf{h}_j^{(s)}, \sum_{l=2}^{L} \mathbf{h}_{j,l}^{(t)})/\tau_2\right)}{\sum_{v_{j'} \in \mathcal{V}} \exp\left(\cos(\mathbf{h}_{j'}^{(s)}, \sum_{l=2}^{L} \mathbf{h}_{j,l}^{(t)})/\tau_2\right)} \quad (8)$$

Adaptive Contrastive Regularization

- **Add a contrastive loss component to the student training process, to "push away" unwanted embeddings.**
- **Specifically, for each node (user/item) in a sample batch, we require that the embedding of that node not only be close to the "anchor" (i.e. the teacher embedding of the same node or the corresponding node) but also be significantly different from other nodes (user-user, user-item, and item-item).**



$$\mathcal{L}_3 = \sum_{u_i \in \mathcal{T}_2} \varphi(u_i, \mathcal{U}, \omega_i) + \varphi(u_i, \mathcal{V}, \omega_i) + \sum_{v_j \in \mathcal{T}_2} \varphi(v_j, \mathcal{V}, \omega_j)$$

$$\varphi(u_i, \mathcal{U}, \omega_i) = \omega_i \cdot \log \sum_{u_{i'} \in \mathcal{U}} \exp(\mathbf{h}_i^{(s)\top} \mathbf{h}_{i'}^{(s)} / \tau_3) \qquad (9)$$

Student Loss Function

$$\mathcal{L}^{(s)} = \mathcal{L}_{\text{rec}} + \lambda_1 \cdot \mathcal{L}_1 + \lambda_2 \cdot \mathcal{L}_2 + \lambda_3 \cdot \mathcal{L}_3 + \lambda_4 \cdot \mathcal{L}_4$$

$$\mathcal{L}_{\text{rec}} = -\sum_{(u_i,v_j)\in\mathcal{T}_2} y_{i,j}, \qquad \mathcal{L}_4 = \|\bar{\mathbf{H}}^{(s)}\|_{\mathrm{F}}^2$$

ℒ_rec: main loss function for the recommendation problem.
ℒ_1: prediction-level distillation loss (simulates the predicted output from the teacher).
ℒ_2: embedding-level distillation loss (simulates the embedding from the teacher).
ℒ_3: contrastive regularization (avoids over-smoothing by pushing away irrelevant embeddings).
ℒ_4: non the embedding, helps avoid overfitting by making the norm of the student embedding smaller.

Learning Algorithm of SimRec

Algorithm 1: Learning Process of SimRec

Input: User-item interaction matrix $\mathbf{A}$, loss weights and temperature factors $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda^{(t)}, \tau_1, \tau_2, \tau_3$, learning rate η, maximum training epochs E, number of graph iterations L, number of MLP layers L'.

Output: Trained embeddings $\bar{\mathbf{H}}^{(s)}$ and MLP parameters $\mathbf{W}$.

Initialize model parameters $\bar{\mathbf{H}}^{(s)}, \bar{\mathbf{H}}^{(t)}, \mathbf{W}$

Train the GCN teacher model for well-trained $\bar{\mathbf{H}}^{(t)}$ (Eq 11)

for $e = 1$ *to* E **do**

    for *mini-batch* $\mathcal{T}_2$ *drawn from* $\mathcal{E}$ **do**

        Sample a batch of triplet $\mathcal{T}_1$

        Calculate preference difference $z_{i,j,k}^{(s)}, z_{i,j,k}^{(t)}$ for samples in $\mathcal{T}_1$ (Eq 6)

        Compute loss $\mathcal{L}_1$ for prediction-level KD (Eq 7)

        Calculate loss $\mathcal{L}_2$ for embedding-level KD (Eq 8)

        Calculate the adjustment factor ω_i, ω_j for users and items in $\mathcal{T}_2$ (Eq 10)

        Compute loss $\mathcal{L}_3$ for contrastive regularization (Eq 9)

        Calculate $\mathcal{L}_{\text{rec}}$ for recommendation task

        Calculate $\mathcal{L}_4$ for weight-decay regularization

        Calculate overall loss $\mathcal{L}^{(s)}$ for the student (Eq 15)

        for *each parameter* θ *in* $\{\bar{H}^{(s)}, W\}$ **do**

            $\theta = \theta - \eta \cdot \partial \mathcal{L}^{(s)} / \partial \theta$;

        end

    end

end

return all parameters $\bar{\mathbf{H}}^{(s)}, \mathbf{W}$

Further Discussion of SimRec

Advantages:

- No need for graph propagation → fast inference.
- Combine GNN knowledge and MLP efficiency.
- Reduce over-smoothing, noise propagation.

Potential applications:

- Real-time recommendation.
- Limited resource systems (edge/mobile).

Extension points:

- Replace GNN teacher with Transformer.
- Learn multi-teacher distillation.
- Adjust contrastive learning with automatic learning (meta-learning).

Evaluation

Answer the research questions

- RQ1: Overall Performance Comparison
- RQ2: Model Ablation Study
- RQ3: Model Scalability Study
- RQ4: Hyperparameter Study
- RQ5: Over-Smoothing Investigation

Evaluation - Dataset

Table 1: Statistics of the experimental datasets.

Dataset	# Users	# Items	# Interactions	Interaction Density
Gowalla	25,557	19,747	294,983	5.85×10^{-4}
Yelp	42,712	26,822	182,357	1.59×10^{-4}
Amazon	76,469	83,761	966,680	1.51×10^{-4}

Training set, Validation set, Test set Ratio:

70%:5%:25%

Evaluation - Performance Comparison

Table 2: Performance comparison on Gowalla, Yelp, and Amazon datasets in terms of *Recall* and *NDCG*.

Data	Metric	BiasMF	NCF	AutoR	PinSage	STGCN	GCMC	NGCF	GCCF	LightGCN	DGCF	SLRec	NCL	SGL	HCCF	*SimRec*	p-val.
Gowalla	Recall@20	0.0867	0.1019	0.1477	0.1235	0.1574	0.1863	0.1757	0.2012	0.2230	0.2055	0.2001	0.2283	0.2332	0.2293	**0.2434**	$2.1e^{-8}$
	NDCG@20	0.0579	0.0674	0.0690	0.0809	0.1042	0.1151	0.1135	0.1282	0.1433	0.1312	0.1298	0.1478	0.1509	0.1482	**0.1592**	$1.2e^{-9}$
	Recall@40	0.1269	0.1563	0.2511	0.1882	0.2318	0.2627	0.2586	0.2903	0.3181	0.2929	0.2863	0.3232	0.3251	0.3258	**0.3399**	$2.4e^{-8}$
	NDCG@40	0.0695	0.0833	0.0985	0.0994	0.1252	0.1390	0.1367	0.1532	0.1670	0.1555	0.1540	0.1745	0.1780	0.1751	**0.1865**	$1.7e^{-9}$
Yelp	Recall@20	0.0198	0.0304	0.0491	0.0510	0.0562	0.0584	0.0681	0.0742	0.0761	0.0700	0.0665	0.0806	0.0803	0.0789	**0.0823**	$3.7e^{-4}$
	NDCG@20	0.0094	0.0143	0.0222	0.0245	0.0282	0.0280	0.0336	0.0365	0.0373	0.0347	0.0327	0.0402	0.0398	0.0391	**0.0414**	$3.8e^{-5}$
	Recall@40	0.0307	0.0487	0.0692	0.0743	0.0856	0.0891	0.1019	0.1151	0.1175	0.1072	0.1032	0.1230	0.1226	0.1210	**0.1251**	$4.8e^{-3}$
	NDCG@40	0.0120	0.0187	0.0268	0.0315	0.0355	0.0360	0.0419	0.0466	0.0474	0.0437	0.0418	0.0505	0.0502	0.0492	**0.0519**	$2.4e^{-4}$
Amazon	Recall@20	0.0324	0.0367	0.0525	0.0486	0.0583	0.0837	0.0551	0.0772	0.0868	0.0617	0.0742	0.0955	0.0874	0.0885	**0.1067**	$1.1e^{-10}$
	NDCG@20	0.0211	0.0234	0.0318	0.0317	0.0377	0.0579	0.0353	0.0501	0.0571	0.0372	0.0480	0.0623	0.5690	0.0578	**0.0734**	$7.0e^{-12}$
	Recall@40	0.0578	0.0600	0.0826	0.0773	0.0908	0.1196	0.0876	0.1175	0.1285	0.0912	0.1123	0.1409	0.1312	0.1335	**0.1535**	$6.6e^{-10}$
	NDCG@40	0.0293	0.0306	0.0415	0.0402	0.0478	0.0692	0.0454	0.0625	0.0697	0.0468	0.0598	0.0764	0.0704	0.0716	**0.0879**	$2.0e^{-12}$

Evaluation - Model Ablation Study

Table 3: Ablation study on key components of SimRec.

Data		Gowalla		Yelp		Amazon	
Variant		Recall	NDCG	Recall	NDCG	Recall	NDCG
$-\mathcal{L}_1$		0.2180	0.1415	0.0756	0.0377	0.1012	0.0692
$-\mathcal{L}_2$	User	0.2292	0.1493	0.0806	0.0405	0.0998	0.0667
	Item	0.2266	0.1477	0.0808	0.0406	0.0974	0.0649
	Both	0.2222	0.1451	0.0787	0.0399	0.0938	0.0626
$-\mathcal{L}_3$	U-I	0.2330	0.1496	0.0814	0.0410	0.0939	0.0607
	U-U	0.2349	0.1512	0.0811	0.0407	0.0965	0.0634
	I-I	0.2331	0.1514	0.0813	0.0409	0.1009	0.0674
	All	0.2282	0.1480	0.0810	0.0407	0.0933	0.0605
SimRec		**0.2434**	**0.1592**	**0.0823**	**0.0414**	**0.1067**	**0.0734**

Evaluation - Model Ablation Study

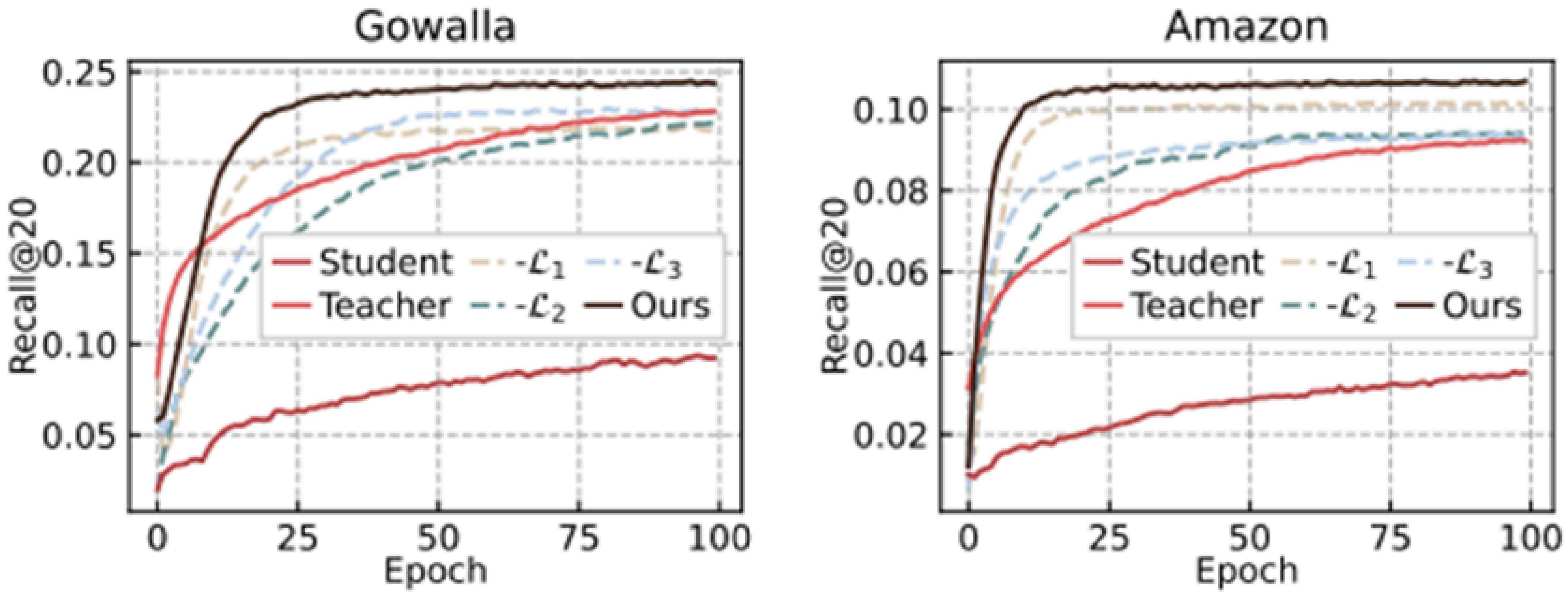


Figure 3: Test performance in each epoch for ablated models.

Evaluation - Model Scalability Study

Table 4: Model performance and per-epoch model inference time of representative methods on large-scale Tmall dataset.

Metric	# Edges	DGCF	SGL	HCCF	NCL	*SimRec*
R@20	1.6M	0.0221	0.0258	0.0272	0.0286	**0.0308**
	2.9M	0.0253	0.0278	0.0283	0.0294	
N@20	1.6M	0.0258	0.0296	0.0309	0.0337	**0.0366**
	2.9M	0.0279	0.0311	0.0319	0.0334	
Time	1.6M	7190.2s	1331.8s	1342.5s	1392.2s	**785.1s**
	2.9M	11431.8s	1456.3s	1530.8s	1693.8s	

Evaluation - Hyperparameter study

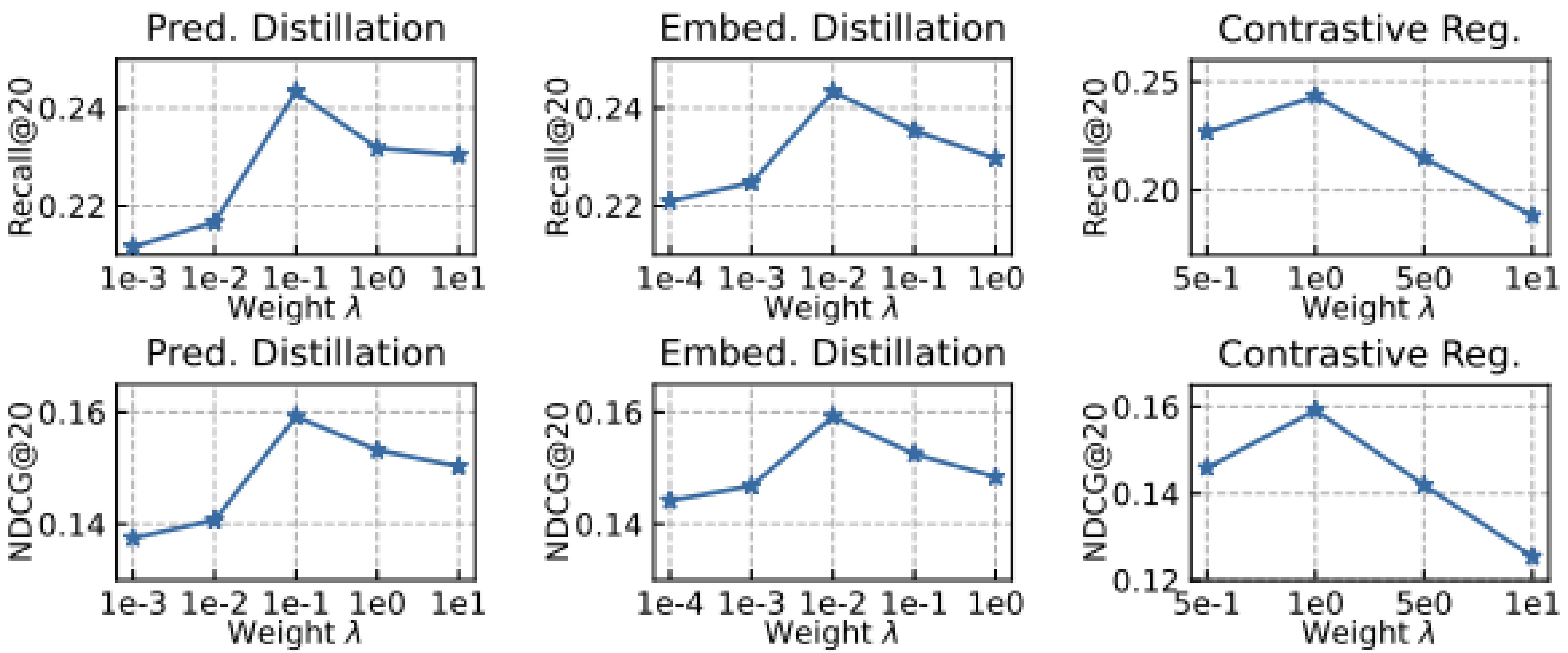
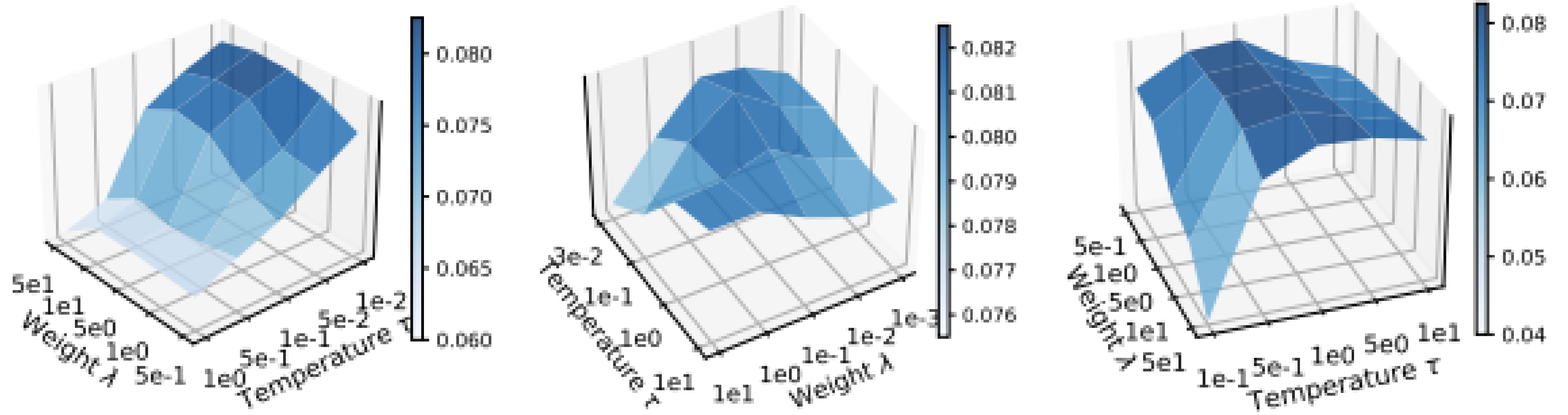


Figure 4: Hyperparameter study for our SimRec model on Gowalla dataset, in terms of *Recall@20* and *NDCG@20*.

Evaluation - Hyperparameter study



(a) Pred. Distillation (b) Embed. Distillation (c) Contrastive Reg.

Figure 5: Impact of weights and temperature in different learning objectives on Yelp, in terms of *Recall@20*.

Evaluation - Hyperparameter study

Table 5: Investigation on the impact of batch size in the prediction-oriented distillation of the proposed SimRec.

Data	Metric	Batch Size $\lvert\mathcal{T}_1\rvert$ in Prediction-Level Distillation					
		$1e3$	$5e3$	$1e4$	$5e4$	$1e5$	$5e5$
Gowalla	Recall	0.2208	0.2361	0.2399	0.2420	0.2434	0.2448
	NDCG	0.1441	0.1530	0.1554	0.1577	0.1592	0.1597
Yelp	Recall	0.0443	0.0730	0.0773	0.0802	0.0823	0.0822
	NDCG	0.0210	0.0372	0.0392	0.0407	0.0414	0.0414

Evaluation - Over-Smoothing Investigation

Table 6: Investigation on the ability to address the over-smoothing effect on Gowalla and Yelp data in terms of MAD.

Data		GCCF	LightGCN	SGL	NCL	HCCF	*SimRec*
Gowalla	User	0.8276	0.8203	0.8412	0.8088	0.8394	**0.8576**
	Item	0.7579	0.7614	0.7702	0.8169	0.7905	**0.8335**
Yelp	User	0.9226	0.9610	0.9755	0.9640	0.9749	**0.9819**
	Item	0.6288	0.7095	0.7191	0.6953	0.6246	**0.7662**

Conclusion

SimRec is a non-GNN recommender model that still achieves high performance thanks to:

- Two-layer distillation strategy (from GNN teacher to MLP student)
- Adaptive contrastive regularization to prevent over-smoothing

Notable results:

- Outperforms GNN and SSL models on many real-world datasets (Gowalla, Yelp, Amazon, Tmall)
- Fast inference speed, no need for graph sampling
- Student model can surpass the GNN model in accuracy

Practical significance:

- Suitable for large-scale recommender systems or resource-constrained devices
- Opens a new direction for graph-independent recommender models

Thank You

For your attention