Ensemble Modeling with Contrastive Knowledge Distillation for Sequential Recommendation

Hanwen Du
hwdu@stu.suda.edu.cn
Soochow University
Suzhou, Jiangsu, China

Huanhuan Yuan
hhyuan@stu.suda.edu.cn
Soochow University
Suzhou, Jiangsu, China

Pengpeng Zhao*
ppzhao@suda.edu.cn
Soochow University
Suzhou, Jiangsu, China

Fuzhen Zhuang
zhuangfuzhen@buaa.edu.cn
Beihang University
Beijing, China

Guanfeng Liu
guanfeng.liu@mq.edu.au
Macquarie University
Sydney, Australia

Lei Zhao
zhaol@suda.edu.cn
Soochow University
Suzhou, Jiangsu, China

Victor S. Sheng
victor.sheng@ttu.edu
Texas Tech University
Lubbock, Texas, USA

ABSTRACT

Sequential recommendation aims to capture users' dynamic interest and predicts the next item of users' preference. Most sequential recommendation methods use a deep neural network as sequence encoder to generate user and item representations. Existing works mainly center upon designing a stronger sequence encoder. However, few attempts have been made with training an ensemble of networks as sequence encoders, which is more powerful than a single network because an ensemble of parallel networks can yield diverse prediction results and hence better accuracy. In this paper, we present **E**nsemble **M**odeling with contrastive **K**nowledge **D**istillation for sequential recommendation (**EMKD**). Our framework adopts multiple parallel networks as an ensemble of sequence encoders and recommends items based on the output distributions of all these networks. To facilitate knowledge transfer between parallel networks, we propose a novel contrastive knowledge distillation approach, which performs knowledge transfer from the representation level via Intra-network Contrastive Learning (ICL) and Cross-network Contrastive Learning (CCL), as well as Knowledge Distillation (KD) from the logits level via minimizing the Kullback-Leibler divergence between the output distributions of the teacher network and the student network. To leverage contextual information, we train the primary masked item prediction task alongside the auxiliary attribute prediction task as a multi-task learning scheme. Extensive experiments on public benchmark datasets show that EMKD achieves a significant improvement compared with the state-of-the-art methods. Besides, we demonstrate that our ensemble method is a generalized approach that can also improve the performance of other sequential recommenders. Our code is available at this link: https://github.com/hw-du/EMKD.

*Corresponding author.

SIGIR '23, Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23)
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9408-6/23/07…$15.00
https://doi.org/10.1145/3539618.3591679

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Sequential Recommendation; Contrastive Learning; Knowledge Distillation

ACM Reference Format:
Hanwen Du, Huanhuan Yuan, Pengpeng Zhao*, Fuzhen Zhuang, Guanfeng Liu, Lei Zhao, and Victor S. Sheng. 2023. Ensemble Modeling with Contrastive Knowledge Distillation for Sequential Recommendation. In ,. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3539618.3591679

1 INTRODUCTION

Recommender systems predict users' interest based on historical interactions. Due to the evolving nature of a user's interest, the task of sequential recommendation views historical interactions as a sequence and aims to model the users' dynamic interest. A typical sequential recommendation model captures such dynamic interest via a sequence encoder, which is a deep neural network that generates the hidden representations of users and items. Recommendations are made based on the information encoded in the hidden representations. Various types of deep neural networks have been shown as effective sequence encoders in capturing the users' dynamic interest, such as Recurrent Neural Network (RNN) [15], Convolutional Neural Network (CNN) [32], unidirectional Transformer [18], and bidirectional Transformer [31].

To accurately capture the users' dynamic interest, most existing works try to improve the architecture of a single sequence encoder in the hope that a diversity of behavioral patterns, such

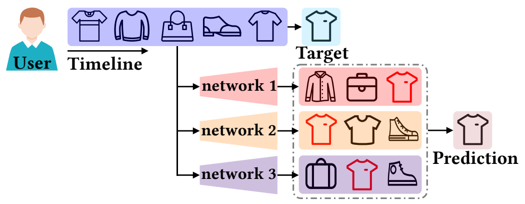


Figure 1: An illustration of ensemble modeling for sequential recommendation. Three parallel networks make different predictions based on users' historical interactions. Although each individual network is unable to make an accurate prediction, combining the predictions of these networks together will get the correct result.

Table 1: Performance comparison (NDCG@10) between the original model and the ensemble models. We independently train two parallel networks initialized with different random seeds and compare the performance with the original model.

Model	GRU4Rec		Caser		SASRec	
	Original	Ensemble(2×)	Original	Ensemble(2×)	Original	Ensemble(2×)
Beauty	0.0175	0.0199	0.0212	0.0247	0.0284	0.0365
Toys	0.0097	0.0102	0.0168	0.0193	0.0320	0.0378
ML-1M	0.0649	0.0720	0.0734	0.0786	0.0918	0.1032

as skip behavior [32] and disordered sequence [31], can be successfully detected by the sequence encoder. However, an alternative approach—training multiple networks as an ensemble of encoders—is less explored in sequential recommendation. A single deep neural network can only converge to a local minimum, and the number of possible local minima grows exponentially with the number of parameters [19]. Therefore, it is hardly possible that two networks with different initializations will converge to the same local minimum even if they share the same architecture [17]. Such a property can bring benefits, because networks converging to different local minima will make different predictions. Although each prediction can only achieve a certain level of accuracy rate, a diversity of predictions combined together can significantly improve the overall accuracy rate. For example, in Figure 1, parallel networks with different initializations will generate a different list of candidate items as the prediction results. It might be hard to get the correct result if we only consider the prediction from one single network. But if we consider predictions from all these networks and choose the most popular item, it will be much easier to get the correct result.

A simple method for ensemble modeling is to train multiple parallel networks independently and average their logits as the final output. We demonstrate its result in Table 1. As we can see, the simple ensemble method indeed improves performance. But due to the absence of knowledge transfer between parallel networks, sometimes its performance margin can be trivial. Some works [22, 29, 44] have shown that an ensemble of networks can benefit from collaborative knowledge distillation, where each individual network acquires supplementary information from peer networks while sharing its own knowledge with others. Compared with independent training, such a collaborative paradigm can improve the performance of each individual network, where knowledge is effectively shared and transferred rather than learned alone. Therefore, it is necessary to design proper techniques for ensemble modeling methods that facilitate knowledge transfer between parallel networks.

Based on the above observations, we present **E**nsemble **M**odeling with contrastive **K**nowledge **D**istillation for sequential recommendation (EMKD). Our framework adopts multiple parallel networks as an ensemble of sequence encoders, and each parallel network is a bidirectional Transformer sequence encoder. To facilitate knowledge transfer between parallel networks, we propose a novel contrastive knowledge distillation approach, which distills knowledge from both the representation level and the logits level. At the representation level, two contrastive learning objectives are designed from both the intra-network perspective and cross-network perspective to distill representational knowledge. At the logits level, a collaborative knowledge distillation approach, which treats each parallel network as both the teacher and the student network, is designed to distill logits-level knowledge. For recommendation training, we introduce masked item prediction as the main task. Since attribute information incorporates rich contextual data that are useful for sequential recommendation [43, 45], we design attribute prediction as an auxiliary task to fuse attribute information.

To verify the effectiveness of our framework, we conduct extensive experiments on public benchmark datasets and show that our framework achieves significant performance improvement compared with the state-of-the-art methods. Furthermore, we show that our method for ensemble modeling is a generalized approach that can be adapted to other types of sequence encoders, such as RNN, CNN, and Transformer, to improve their performances. Since training efficiency can be a major concern for ensemble methods, we compare the training speed and convergence speed of our framework with other state-of-the-art methods and find out that our framework only shows a minor reduction in training efficiency. For these reasons, we think that it is worthwhile to adopt ensemble modeling methods in sequential recommendation. Our contributions can be summarized as follows:

- We propose a novel framework called Ensemble Modeling with Contrastive Knowledge Distillation for sequential recommendation (EMKD). To the best of our knowledge, this is the first work to apply the ensemble modeling to sequential recommendation.
- We propose a novel contrastive knowledge distillation approach that facilitates knowledge transfer and distills knowledge from both the representation level and the logits level.
- We conduct extensive experiments on public benchmark datasets and show that our framework can significantly outperform the state-of-the-art methods. Our framework can also be adapted to other sequential recommenders to improve their performances.

2 RELATED WORK

2.1 Sequential Recommendation

Traditional methods for sequential recommendation adopt the Markov Chains (MCs) to model item transitions, such as MDP [28], FPMC [27] and Fossil [14]. With the advancements in deep learning, RNN-based model [15] and CNN-based model [32] have all been shown as effective sequence encoders. The recent success of Transformer [36] also motivates the designing of Transformer-based sequence encoders. For example, SASRec [18] adopts unidirectional Transformer to automatically assign attention weights to different items, BERT4Rec [31] adopts bidirectional Transformer with the cloze task [6] to adapt for interactions that do not follow a rigid sequential order. Some works also leverage the attribute information about items in order to provide contextual data for sequential recommendation. For example, FDSA [43] adopts feature-level self-attention blocks to fuse attribute data into sequences, S^3-Rec [45] proposes self-supervised objectives to model the correlations between items, sequences and attributes, MMInfoRec [25] combines the attribute encoder with a memory module and a sampling strategy to capture long-term preferences. Different from these works that only focus on a single encoder, our framework trains an ensemble of networks with contrastive knowledge distillation.

2.2 Ensemble Modeling and Knowledge Distillation

Ensemble modeling [10, 17, 22, 29, 30, 34, 44] has long been established as an effective approach for improving the accuracy and robustness of neural networks. One commonly adopted technique is Dropout [30], which creates a self-ensemble of networks by randomly dropping neurons at the training stage. As for the ensemble of multiple parallel networks, some works have recognized the necessity of facilitating knowledge when training multiple parallel networks, which is often implemented via knowledge distillation [16]. For example, DML [44] proposes a collaborative learning strategy by distilling knowledge between parallel networks, ONE [22] proposes an on-the-fly native ensemble method for one-stage online distillation. However, these methods are all proposed for the compression of large-scale models on computer vision tasks, and how to design efficient and effective methods to facilitate knowledge transfer for ensemble sequential models remains to be explored.

2.3 Contrastive Learning

Contrastive learning has achieved great success in computer vision [1–5, 8, 13, 33, 35], natural language processing [9, 41], and recommender systems [26, 39, 40, 42]. In sequential recommendation, contrastive learning has also been shown as an effective approach for alleviating data sparsity and learning better representations. For example, CL4SRec [40] first introduces contrastive learning into sequential recommendation by proposing three data augmentation operators, DuoRec [26] performs contrastive learning from the model level to mitigate the representation degeneration problem, CML [39] combines contrastive learning with meta learning for personalized multi-behavior recommendation. Different from these works, our framework formulates contrastive learning as a way of contrastive representation distillation [33] to facilitate knowledge transfer between the ensemble networks.

3 PROPOSED FRAMEWORK

In this section, we introduce the architecture of our proposed framework, Ensemble Modeling with contrastive Knowledge Distillation (EMKD). The architecture of EMKD is illustrated in Figure 2. Our framework trains an ensemble of sequence encoders and averages the logits of these sequence encoders for inference. We design masked item prediction as the primary task for recommendation training and attribute prediction as the auxiliary task for attribute information fusion. To facilitate knowledge transfer between these parallel networks, we propose a novel approach for contrastive knowledge distillation, which performs contrastive representation distillation from the representation level and knowledge distillation from the logits level.

3.1 Problem Statement

In sequential recommendation, we denote $\mathcal{U}$ as a set of users, $\mathcal{V}$ as a set of items, and $\mathcal{A}$ as a set of attributes. Each user $u \in \mathcal{U}$ is associated with an interaction sequence $s_u = [v_1, v_2, \cdots, v_t, \cdots, v_T]$ sorted in the chronological order, where $v_t \in \mathcal{V}$ denotes the item that user u has interacted with at the t-th timestamp and T indicates the sequence length. Each item v_t has its own attribute set $\alpha_t = \{a_1, a_2, \cdots, a_{|\alpha_t|}\} \subset \mathcal{A}$ as a subset of all the attributes. The task of sequential recommendation is to predict the next item that user u will probably interact with, and it can be formulated as generating the probability distribution over all candidate items for user u at the time step $T + 1$:

$$p(v_{T+1} = v|s_u)$$

3.2 Ensemble Sequence Encoder

A sequential recommendation model encodes essential sequence information via a sequence encoder, which takes user sequences as input and outputs hidden representations of sequences. In our framework, we adopt bidirectional Transformer as the base sequence encoder, which is able to model item correlations from both directions. The architecture of bidirectional Transformer consists of an embedding layer and a bidirectional Transformer module. For the input sequence $s_u = [v_1, v_2, \cdots, v_T]$, an item embedding matrix $\boldsymbol{V} \in \mathbb{R}^{|\mathcal{V}|\times d}$ and a positional embedding matrix $\boldsymbol{P} \in \mathbb{R}^{T\times d}$ and are combined together to represent items in the hidden space, where T denotes the maximum sequence length of our model, d is the hidden dimensionality. The computation of the embedding layer is formulated as follows:

$$\boldsymbol{E}(s_u) = [\boldsymbol{v}_1 + \boldsymbol{p}_1, \boldsymbol{v}_2 + \boldsymbol{p}_2, \cdots, \boldsymbol{v}_T + \boldsymbol{p}_T] \tag{1}$$

We then feed $\boldsymbol{E}(s_u)$ into the bidirectional Transformer module Trm and fetch the output $\boldsymbol{H} \in \mathbb{R}^{T\times d}$ as the hidden representations of the sequence. In general, we denote a sequence encoder $f(\cdot)$ that takes user sequence s_u as the input and outputs the hidden representations $\boldsymbol{H}(s_u)$ of this sequence:

$$\boldsymbol{H}(s_u) = [\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_T] = f(s_u) = Trm(\boldsymbol{E}(s_u)) \tag{2}$$

As we have mentioned above, an ensemble of parallel networks can generate different predictions, which will be helpful for improving the overall accuracy rate. We adopt N parallel networks as an ensemble of sequence encoders. Each parallel network $f^n(\cdot)$ is a bidirectional Transformer sequence encoder, where $1 \le n \le N$. We initialize these parallel networks with different random seeds so that they will generate different prediction results.

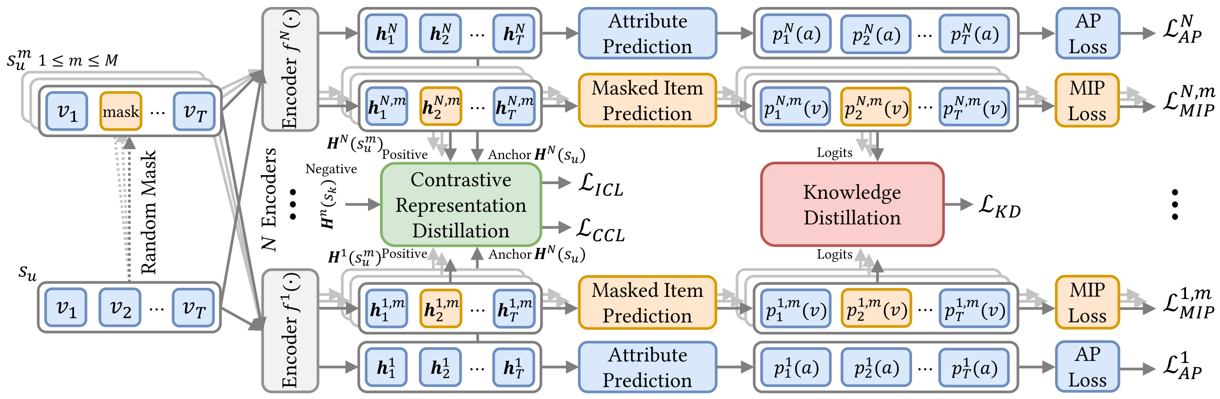


Figure 2: An overview of EMKD with N parallel networks $f^1(\cdot), \cdots, f^N(\cdot)$. For each original sequence s_u, we generate M different masked sequences. The hidden representations of the original sequence $H^1(s_u), \cdots, H^N(s_u)$ serve as the anchor for contrastive representation distillation and are used for the attribute prediction task, while the hidden representations of the masked sequences $H^1(s_u^m), \cdots, H^N(s_u^m)$ serve as positive samples for contrastive representation distillation and are used for the masked item prediction task. Negative samples $H^n(s_k)(1 \le n \le N)$ for contrastive representation distillation are collected from the same batch. We compute the Kullback-Leibler divergence on the logits of the masked item prediction task between different networks for knowledge distillation.

3.3 Multi-Task Learning

We adopt a multi-task learning scheme to enhance sequential recommendation with contextual information. Specifically, we adopt masked item prediction as the main task for recommendation training. An attribute prediction task is also introduced as the auxiliary task to fuse attribute information.

3.3.1 *Masked Item Prediction.*

Since bidirectional Transformers are typically trained with the masked item prediction task, we follow BERT4Rec [31] and adopt masked item prediction as the main task, which requires the model to reconstruct the masked items. Specifically, for each user sequence s_u, we use different random seeds to generate M different masked sequences. In each masked sequence s_u^m, a proportion ρ of items $\mathcal{I}_u^m = (t_1^m, t_2^m, \cdots, t_L^m)$ are randomly replaced with the mask token [mask]. Here $L = \lceil \rho * T \rceil$ and $\mathcal{I}_u^m$ denotes the indices of the masked items. The process of random item masking is defined as follows:

$$s_u^m = \left[\widehat{v}_1, \widehat{v}_2, \cdots, \widehat{v}_T\right], 1 \le m \le M$$
$$\widehat{v}_t = \begin{cases} v_t, & t \notin \mathcal{I}_u^m \\ [\text{mask}], & t \in \mathcal{I}_u^m \end{cases} \tag{3}$$

Each masked sequence s_u^m is then fed into the ensemble networks to generate the corresponding hidden representations:

$$\boldsymbol{H}^n(s_u^m) = [\boldsymbol{h}_1^{n,m}, \boldsymbol{h}_2^{n,m}, \cdots, \boldsymbol{h}_T^{n,m}] = f^n(s_u^m), 1 \le n \le N, 1 \le m \le M \tag{4}$$

A linear layer is adopted as the item classifier to convert the hidden representations into probability distribution over candidate items. Given the output of hidden representation $\boldsymbol{h}_t^{n,m}$ at position t, the computation is formulated as follows:

$$p_t^{n,m}(v) = \boldsymbol{h}_t^{n,m}\boldsymbol{W} + \boldsymbol{b}, 1 \le t \le T \tag{5}$$

where $\boldsymbol{W} \in \mathbb{R}^{d \times |\mathcal{V}|}$ is the weight matrix and $\boldsymbol{b} \in \mathbb{R}^{|\mathcal{V}|}$ is the bias term. Finally, we calculate the cross-entropy loss for the m-th masked sequence from the n-th network:

$$q_t^{n,m}(v_i) = \frac{\exp(p_t^{n,m}(v_i))}{\sum_{j=1}^{|\mathcal{V}|} \exp(p_t^{n,m}(v_j))}$$
$$\mathcal{L}_{MIP}^{n,m} = -\sum_{t \in \mathcal{I}_u^m} \sum_{i=1}^{|\mathcal{V}|} y_i \log q_t^{n,m}(v_i) \tag{6}$$

where $y_i = 1$ if $v_i = v_t$ (the predicted item v_i is the ground-truth item v_t) else 0. Note that only the masked items within the sequence are considered when calculating the loss function for the masked item prediction task. Since we have N parallel networks where each network is fed with M different masked sequences, we sum up each $\mathcal{L}_{MIP}^{n,m}$ as the total loss function of the masked item prediction task:

$$\mathcal{L}_{MIP} = \sum_{n=1}^{N} \sum_{m=1}^{M} \mathcal{L}_{MIP}^{n,m} \tag{7}$$

3.3.2 *Attribute Prediction.*

Attribute provides fine-grained features of items, which is a readily-available source of auxiliary information for sequential recommendation. Some works [25, 43, 45] have shown that incorporating attribute information benefits the main task of sequential recommendation. Therefore, we propose an auxiliary attribute prediction task to incorporate attribute information. Different from previous works, we do not construct an explicit attribute embedding to model item-attribute correlations. Instead, we treat attribute prediction as a multi-label classification task that requires to the model to classify an item under the correct attribute(s). Specifically, we feed the original sequence s_u into parallel networks to generate the hidden representations:

$$\boldsymbol{H}^n(s_u) = [\boldsymbol{h}_1^n, \boldsymbol{h}_2^n, \cdots, \boldsymbol{h}_T^n] = f^n(s_u), 1 \le n \le N \tag{8}$$

We then adopt another linear layer with the sigmoid activation function as the attribute classifier to convert the hidden representation $\boldsymbol{h}_t^n$ at position t into probability distribution over the attributes:

$$p_t^n(a) = \sigma(\boldsymbol{h}_t^n \boldsymbol{W}' + \boldsymbol{b}'), 1 \le t \le T \tag{9}$$

where $\boldsymbol{W}' \in \mathbb{R}^{d \times |\mathcal{A}|}$ is the weight matrix, $\boldsymbol{b}' \in \mathbb{R}^{|\mathcal{A}|}$ is the bias term, and σ is the sigmoid function. Given that an item may be associated with multiple attributes, we compute the binary cross-entropy loss function for the attribute prediction task from the n-th network:

$$\mathcal{L}_{AP}^n = -\sum_{t=1}^{T} \sum_{i=1}^{|\mathcal{A}|} \left[y_i \log p_t^n(a_i) + (1 - y_i) \log(1 - p_t^n(a_i)) \right] \tag{10}$$

where $y_i = 1$ if $a_i \in \alpha_t$ (the predicted attribute a_i is a correct attribute in the associated attribute set α_t of item v_t) else 0. Each $\mathcal{L}_{AP}^n$ is then summed up as the total loss function of the attribute prediction task:

$$\mathcal{L}_{AP} = \sum_{n=1}^{N} \mathcal{L}_{AP}^n \tag{11}$$

3.4 Contrastive Knowledge Distillation

To transfer knowledge between parallel networks, we design a contrastive knowledge distillation approach that transfers both the representational knowledge and the logits-level knowledge.

3.4.1 Contrastive Representation Distillation. Contrastive learning aims to align the positive sample (i.e., a data-augmented sequence) with its anchor (i.e., an original sequence) and push negative samples apart from the anchor. In our framework, the hidden representation $\boldsymbol{H}^n(s_u)$ from the attribute prediction task can be viewed as an anchor representation because it is obtained from the original user sequence s_u. Besides, the other hidden representations $\boldsymbol{H}^n(s_u^m)$ obtained from the masked item prediction task can be viewed as positive samples, because they are obtained from masked user sequences s_u^m, which is a form of data augmentation. Similar to the in-batch negative sampling strategy in [2, 26, 40], the other user sequences $\{s_k\}_{k=1,k \ne u}^{\mathcal{B}}$ from the same batch can be viewed as negative samples, where $\mathcal{B}$ denotes the batch size.

Our method for contrastive representation distillation contains two objectives. First, an Intra-network Contrastive Learning (ICL) objective is designed to contrast the hidden representations of different masked sequences outputted by the same network. The loss function of ICL for the n-th network is formulated as follows:

$$\mathcal{L}_{ICL}^n = -\sum_{m=1}^{M} \log \frac{\exp(g(\boldsymbol{H}^n(s_u), \boldsymbol{H}^n(s_u^m))/\tau)}{\sum_{k=1,k \ne u}^{\mathcal{B}} \exp(g(\boldsymbol{H}^n(s_u), \boldsymbol{H}^n(s_k))/\tau)} \tag{12}$$

where $g(\cdot)$ is the cosine similarity function, τ is a temperature hyper-parameter, M is the number of the masked sequences. Each $\mathcal{L}_{ICL}^n$ is then summed up as the total loss function for ICL:

$$\mathcal{L}_{ICL} = \sum_{n=1}^{N} \mathcal{L}_{ICL}^n \tag{13}$$

ICL alone cannot effectively transfer knowledge between the parallel networks, because the representations from different networks are not contrasted against each other. To transfer representational knowledge between the parallel networks, a Cross-network Contrastive Learning (CCL) objective is introduced to contrast the hidden representations of different masked sequences outputted by different networks. Similar to Eq. 12, the loss function of CCL between the x-th network and the y-th network is formulated as follows:

$$\mathcal{L}_{CCL}^{x,y} = -\sum_{m=1}^{M} \log \frac{\exp(g(\boldsymbol{H}^x(s_u), \boldsymbol{H}^y(s_u^m))/\tau)}{\sum_{k=1,k \ne u}^{\mathcal{B}} \exp(g(\boldsymbol{H}^x(s_u), \boldsymbol{H}^y(s_k))/\tau)} \tag{14}$$

where $x, y \in [1, N]$ are the indices of the networks. Since we have N parallel networks, permutations of $\mathcal{L}_{CCL}^{x,y}$ are summed up as the total loss function for CCL:

$$\mathcal{L}_{CCL} = \sum_{x=1}^{N} \sum_{y=1, y \ne x}^{N} \mathcal{L}_{CCL}^{x,y} \tag{15}$$

3.4.2 Logits-Level Knowledge Distillation. The task of sequential recommendation takes the logits $p_t^{n,m}(v)$ as the final prediction result. Although representational knowledge is distilled through contrastive learning, we still need Knowledge Distillation (KD) from the logits level so as to transfer knowledge more related with the sequential recommendation task. Following [16, 22] for knowledge distillation, we minimize the Kullback-Leibler divergence between the teacher logits $p_t^{x,m}(v)$ and the student logits $p_t^{y,m}(v)$. Specifically, we first compute the softmax probability distribution at temperature τ:

$$z_t^{x,m}(v_i) = \frac{\exp(p_t^{x,m}(v_i)/\tau)}{\sum_{j=1}^{|\mathcal{V}|} \exp(p_t^{x,m}(v_j)/\tau)}, z_t^{y,m}(v_i) = \frac{\exp(p_t^{y,m}(v_i)/\tau)}{\sum_{j=1}^{|\mathcal{V}|} \exp(p_t^{y,m}(v_j)/\tau)} \tag{16}$$

The computation of the Kullback-Leibler divergence between the x-th network and the y-th network can then be formulated as follows:

$$\mathcal{L}_{KD}^{x,y} = \sum_{m=1}^{M} \sum_{t \in \mathcal{I}_u^m} \sum_{i=1}^{|\mathcal{V}|} z_t^{x,m}(v_i) \log \frac{z_t^{x,m}(v_i)}{z_t^{y,m}(v_i)} \tag{17}$$

where the probability distributions of the M masked sequences are all considered for knowledge distillation. Note that when $z_t^{x,m}(v_i)$ is treated as the teacher logits, we cut off its gradient for stability.

The logits-level knowledge is distilled in a collaborative style, where each network can simultaneously be the student who acquires knowledge from other peer networks and the teacher who provides knowledge for other peer networks. In this way, the distilled knowledge can be effectively shared and transferred between all parallel networks. Therefore, we sum up the permutations of $\mathcal{L}_{KD}^{x,y}$ for all the N parallel networks to perform a collaborative paradigm of knowledge distillation:

$$\mathcal{L}_{KD} = \sum_{x=1}^{N} \sum_{y=1, y \ne x}^{N} \mathcal{L}_{KD}^{x,y} \tag{18}$$

3.5 Training and Inference

3.5.1 Overall Training Objective. We sum up the loss function for the masked item prediction task, the attribute prediction task, the ICL objective, the CCL objective, and the KD objective as the total loss function to jointly optimize our framework:

$$\mathcal{L}_{EMKD} = \mathcal{L}_{MIP} + \mathcal{L}_{AP} + \lambda(\mathcal{L}_{ICL} + \mathcal{L}_{CCL}) + \mu \mathcal{L}_{KD} \tag{19}$$

where λ, μ are weight hyper-parameters.

3.5.2 Inference. To infer the next item for user u, we append the mask token to the end of the sequence s_u to obtain $s_{u'} = [v_2, v_3 \cdots, v_T, [\text{mask}]]$[1]. The hidden representations corresponding to the mask token are converted into probability distributions by the item classifier defined in Equation 5. Predictions from all the parallel networks are averaged as the final prediction result:

$$\begin{aligned} \boldsymbol{H}^n(s_{u'}) &= [\boldsymbol{h}_2^n, \boldsymbol{h}_3^n, \cdots, \boldsymbol{h}_T^n, \boldsymbol{h}_{\text{mask}}^n] = f^n(s_{u'}) \\ p(v) &= \frac{1}{N} \sum_{n=1}^{N} (\boldsymbol{h}_{\text{mask}}^n \boldsymbol{W} + \boldsymbol{b}) \end{aligned} \tag{20}$$

3.6 Discussion

One possible reason why ensemble methods are superior for sequential recommendation is that it can handle behavior uncertainty. Users' sequential behaviors are uncertain rather than deterministic, and users' preferences may shift under different circumstances [7]. For example, a user favours movies with Action and Comedy genres. However, there are many candidate movies that fall under these genres, and the user may choose different movies under different situations. In such cases, the diversity of prediction results brought by ensemble methods can consider all these situations and generate a more accurate recommendation result.

Another possible reason is that ensemble methods can mitigate the adverse effects of noisy data. The datasets for sequential recommendation are usually sparse and noisy. In such cases, the knowledge transfer mechanisms between the ensemble networks can filter out the incorrect predictions introduced by noisy labels and transfer the useful knowledge to the whole network [11], thus improving the overall accuracy rate and model robustness.

4 EXPERIMENT

In this section, we present the details of our experiments and answer the following research questions (**RQs**):

- **RQ1**: How does EMKD perform comparing with other state-of-the-art methods for sequential recommendation?
- **RQ2**: What are the influences of different hyper-parameters in EMKD?
- **RQ3**: What is the effectiveness of different components in EMKD?
- **RQ4**: Can we adapt EMKD to other sequential recommenders to improve their performances?
- **RQ5**: Compared with other state-of-the-art methods, will ensemble modeling significantly reduce training efficiency?
- **RQ6**: Does the good performance of EMKD result from the proposed knowledge transfer mechanisms between the ensemble networks or does it simply come from the increase in the parameter size?

4.1 Settings

4.1.1 Dataset. We conduct experiments on three public benchmark datasets collected from two platforms. The Amazon dataset [23] contains users' reviews on varying categories of products. We select two subcategories, **Beauty** and **Toys**, and use the fine-grained categories and the brands of the products as attributes. Another dataset MovieLens-1M (**ML-1M**) [12] contains users' ratings on movies, and we use the genres of the movies as attributes.

[1]We drop the first item v_1 due to the restriction of maximum sequence length.

Table 2: Dataset statistics after preprocessing.

Datasets	Beauty	Toys	ML-1M
#users	22,363	19,412	6,040
#items	12,101	11,924	3,953
#actions	198,502	167,597	1,000,209
avg. actions/user	8.9	8.6	163.5
avg. actions/item	16.4	14.1	253.0
sparsity	99.93%	99.93%	95.81%
#attributes	1,221	1,027	18
avg. attributes/item	5.1	4.3	1.7

Following [18, 26, 31, 40, 45], we treat all interactions as implicit feedbacks. To construct user sequences, we remove duplicated interactions and sort each user's interactions by their timestamps. Users related with less than 5 interactions and items related with less than 5 users are filtered out from the dataset. The processed dataset statistics are presented in Table 2.

4.1.2 Metrics. We choose top-K Hit Ratio (HR@K) and Normalized Discounted Cumulative Gain (NDCG@K) with $K \in \{5, 10\}$ as metrics to evaluate the performance. The *leave-one-out* evaluation strategy is adopted, holding out the last item for test, the second-to-last item for validation, and the rest for training. Since sampled metrics might lead to unfair comparisons [21], we rank the prediction results on the whole item set without sampling.

4.1.3 Baselines. We include baseline methods from three groups for comparison:

- **General sequential methods** utilize a sequence encoder to generate the hidden representations of users and items. For example, GRU4Rec [15] adopts RNN as the sequence encoder, Caser [32] adopts CNN as the sequence encoder, SASRec [18] adopts unidirectional Transformer as the sequence encoder, BERT4Rec [31] adopts bidirectional Transformer as the sequence encoder.
- **Attribute-aware sequential methods** fuse attribute information into sequential recommendation. For example, FDSA [43] applies self-attention blocks to capture transition patterns of items and attributes, S^3-Rec [45] proposes self-supervised objectives to model the correlations between items, sequences and attributes, MMInfoRec [25] augments the attribute encoder with a memory module and a multi-instance sampling strategy.
- **Contrastive sequential methods** design auxiliary objectives for contrastive learning based on general sequential methods. For example, CL4SRec [40] proposes data augmentation strategies for contrastive learning in sequential recommendation, DuoRec [26] proposes both supervised and unsupervised sampling strategies for contrastive learning in sequential recommendation.

4.1.4 Implementation. The codes for GRU4Rec, Caser, SASRec, BERT4Rec, S^3-Rec, MMInfoRec, and DuoRec are provided by their authors. We implement FDSA and CL4SRec in PyTorch [24]. We follow the instructions from the original papers to set and tune the hyper-parameters.

We also implement our framework in PyTorch [24]. We set the number of Transformer blocks and attention heads as 2, batch size as 256. Following BERT [6] we set the mask proportion ρ as 0.15. For other hyper-parameters, we tune λ within $[0.01, 1]$, μ within $[0.01, 1]$, τ within $[0.1, 10]$, the number of masked sequences M within $[1, 8]$. The default setting for the number of parallel networks N is 3, and variants are studied in Section 4.4. The default setting for the hidden dimensionality d is set as 256, and model robustness w.r.t different hidden dimensionality sizes is studied in Section 4.3.4. We use an Adam optimizer [20] with an initial learning rate 0.001, and the learning rate decays exponentially after every 100 epochs. We train our model for 250 epochs and select the checkpoint with the best validation result for test.

4.2 Overall Performance Comparison (RQ1)

Table 3 presents the overall performance of EMKD and baseline methods. Based on the experiment results, we can see:

- EMKD outperforms all baseline methods on both sparse and dense datasets with the relative performance improvements ranging from 13.05% to 33.48%. We attribute the performance improvement to these factors: (1) ensemble modeling uses multiple parallel networks to yield diverse predictions, and combining a diversity of prediction results is more accurate than the prediction of a single network; (2) contrastive knowledge distillation facilitates knowledge transfer between parallel networks, and each individual network benefits from this collaborative learning paradigm; (3) the attribute information provides rich contextual data and can benefit the main task of sequential recommendation.
- Among general sequential methods, Transformer-based sequence encoders (e.g., SASRec, BERT4Rec) outperform RNN-based (e.g., GRU4Rec) or CNN-based (e.g., Caser) sequence encoders. This suggests that the self-attention mechanism can effectively model sequential patterns. Moreover, attribute-aware sequential methods outperform general sequential methods owing to the incorporation of attribute information. Besides, contrastive sequential methods show performance improvements compared with general sequential methods. This is probably because contrastive learning serves as a regularization objective that can alleviate the data sparsity issue and improve model robustness.

4.3 Hyper-parameter Sensitivity (RQ2)

In this section, we study the influences of important hyper-parameters, including the number of masked sequences M, temperature τ, weight hyper-parameters λ and μ, hidden dimensionality d. To control variables we only change one hyper-parameter at one time while keeping the other hyper-parameters optimal.

4.3.1 Number of Masked Sequences. M regulates how many masked sequences are available for training. From Figure 3a, we can see that the performance increases as more masked sequences are available. This is because masking different items in the same sequence brings various semantic patterns, and reconstructing all these items will help the model gain more knowledge. However, such performance gain also shows an upper limit for multiple masked sequences. With sufficient masked sequences, the performance hardly improves even if we further increase M.

4.3.2 Temperature. Temperature τ regulates the hardness of the probability distributions for contrastive representation distillation and knowledge distillation. From Figure 3b, we can see that an appropriate choice of τ should be neither too large nor too small. This is because a smaller τ produces harder probability distributions that makes the model intolerant to semantically similar samples, while a larger τ produces softer probability distributions that makes the model insensitive to semantic differences [37, 38].

4.3.3 Weight Hyper-parameters. λ and μ are two weight hyper-parameters that control the strength of contrastive representation distillation and knowledge distillation. From Figure 3c and Figure 3d, we can see that a proper choice of weight hyper-parameters can significantly improve the performance, while selecting weight hyper-parameters that are either too large or too small will undermine the performance.

4.3.4 Hidden Dimensionality d. Figure 3e reports the performances w.r.t different hidden dimensionality d. We can see that the performance steadily increases as the hidden dimensionality becomes larger, and the best performance is reached when $d = 256$ (default setting). However, the performance slightly drops when $d = 512$, which is probably due to overfitting.

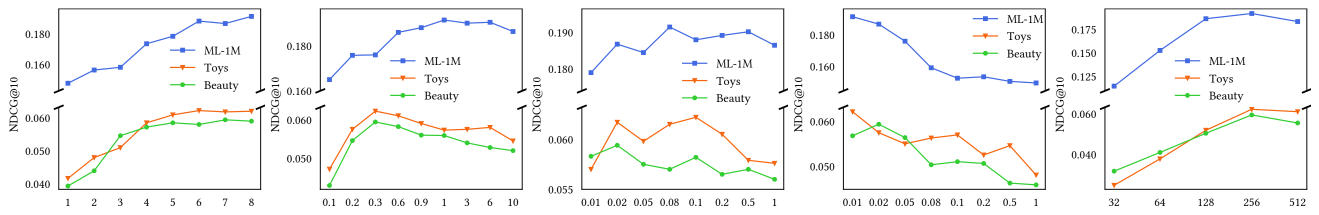
4.4 Ablation Study (RQ3)

In this section, we conduct an ablation study to investigate the effectiveness of the key components in our framework. Table 4 presents the performance of EMKD under the default setting and its variants. Based on the experiment results, we can see that:

- **Ensemble networks outperform a single network.** We can see that (6) shows the worst performance compared with other ensemble methods. This observation verifies our claim that ensemble modeling is more powerful than a single network.
- **Ensemble networks learn better with effective knowledge transfer.** Our method for contrastive knowledge distillation contains three objectives—ICL, CCL and KD. From (2)-(4) we can see that each objective plays a crucial role in facilitating knowledge transfer between parallel networks, and removing any objective leads to performance decrease. Compared with (5), we can see that ensemble networks with knowledge transfer methods outperform the simple ensemble of independently-trained networks.
- **Training EMKD with 3 parallel networks is the optimal setting.** Increasing the number of parallel networks can improve performance because training more parallel networks can increase the diversity of prediction results. However, we cannot train an infinite number of parallel networks due to the limitation of computational cost. From (1)(6)(7)(8) we can see that training EMKD with 3 parallel networks almost reaches the best performance. Although (8) shows a slight improvement compared with (1), training EMKD with 4 parallel networks is too costly and also increases the risk of overfitting.
- **Sequential recommendation benefits from the auxiliary attribute information.** From (9) we can see that removing the auxiliary task of attribute prediction will hurt performance. This observation is also consistent with previous works [25, 43, 45] that utilizing attribute information can enhance the performance of sequential recommendation models.

Table 3: Overall performance of different methods for sequential recommendation. The best score and the second-best score in each row are bolded and underlined, respectively. The last column indicates improvements over the best baseline method.

Dataset	Metric	GRU4Rec	Caser	SASRec	BERT4Rec	FDSA	S^3-Rec	MMInfoRec	CL4SRec	DuoRec	EMKD	Improv.
Beauty	HR@5	0.0206	0.0254	0.0371	0.0364	0.0317	0.0382	0.0527	0.0396	0.0559	**0.0702**	25.58%
	HR@10	0.0332	0.0436	0.0592	0.0583	0.0496	0.0634	0.0739	0.0630	0.0867	**0.0995**	14.76%
	NDCG@5	0.0139	0.0154	0.0233	0.0228	0.0184	0.0244	0.0378	0.0232	0.0331	**0.0500**	32.28%
	NDCG@10	0.0175	0.0212	0.0284	0.0307	0.0268	0.0335	0.0445	0.0307	0.0430	**0.0594**	33.48%
Toys	HR@5	0.0121	0.0205	0.0429	0.0371	0.0269	0.0440	0.0579	0.0503	0.0539	**0.0745**	28.67%
	HR@10	0.0184	0.0333	0.0652	0.0524	0.0483	0.0705	0.0818	0.0736	0.0744	**0.1016**	24.21%
	NDCG@5	0.0077	0.0125	0.0248	0.0259	0.0227	0.0286	0.0408	0.0264	0.0340	**0.0534**	30.88%
	NDCG@10	0.0097	0.0168	0.0320	0.0309	0.0281	0.0369	0.0484	0.0339	0.0406	**0.0622**	28.51%
ML-1M	HR@5	0.0806	0.0912	0.1078	0.1308	0.0953	0.1128	0.1454	0.1142	0.1930	**0.2315**	19.95%
	HR@10	0.1344	0.1442	0.1810	0.2219	0.1645	0.1969	0.2248	0.1815	0.2865	**0.3239**	13.05%
	NDCG@5	0.0475	0.0565	0.0681	0.0804	0.0597	0.0668	0.0856	0.0705	0.1327	**0.1616**	21.78%
	NDCG@10	0.0649	0.0734	0.0918	0.1097	0.0864	0.0950	0.1203	0.0920	0.1586	**0.1915**	20.74%



(a) Number of Masked Seq. **(b) Temperature τ** **(c) Hyper-parameter λ** **(d) Hyper-parameter μ** **(e) Hidden Dimensionality**

Figure 3: Performance (NDCG@10) comparison w.r.t different hyper-parameters on three datasets.

Table 4: Ablation study (NDCG@10) on three datasets. Bold score indicates the performance under the default setting. ↑ indicates the performance better than the default setting.

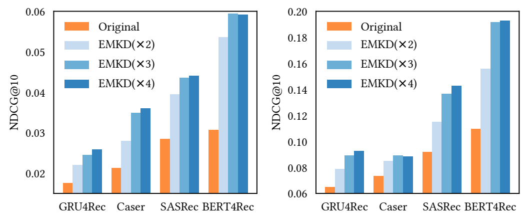
Architecture	Dataset Beauty	Toys	ML-1M
(1) EMKD(×3)	**0.0594**	**0.0622**	**0.1915**
(2) Remove ICL	0.0529	0.0545	0.1679
(3) Remove CCL	0.0552	0.0560	0.1807
(4) Remove KD	0.0537	0.0571	0.1758
(5) Independent Training	0.0452	0.0484	0.1476
(6) Single Encoder	0.0363	0.0375	0.1183
(7) EMKD(×2)	0.0536	0.0568	0.1792
(8) EMKD(×4)	0.0591	0.0629↑	0.1930↑
(9) Remove AP	0.0578	0.0609	0.1831

Table 5: Performance comparison (NDCG@10) of models with different parameter sizes on three datasets. ∗ indicates the default setting for each model.

	Beauty		Toys		ML-1M	
Architecture	Params.	NDCG@10	Params.	NDCG@10	Params.	NDCG@10
SASRec-2 Layers*	4.69M	0.0284	4.65M	0.0320	2.51M	0.0918
SASRec-4 Layers	6.27M	0.0301	6.23M	0.0313	4.09M	0.0896
SASRec-6 Layers	7.85M	0.0298	7.80M	0.0332	5.67M	0.0857
SASRec-8 Layers	9.43M	0.0279	9.38M	0.0305	7.24M	0.0932
SASRec-10 Layers	11.01M	0.0282	10.96M	0.0310	8.82M	0.0881
BERT4Rec-2 Layers*	7.80M	0.0307	7.71M	0.0309	3.53M	0.1097
BERT4Rec-4 Layers	9.38M	0.0328	9.29M	0.0312	5.11M	0.1113
BERT4Rec-6 Layers	10.96M	0.0332	10.87M	0.0306	6.69M	0.1100
BERT4Rec-8 Layers	12.54M	0.0310	12.45M	0.0298	8.27M	0.1093
BERT4Rec-10 Layers	14.12M	0.0319	14.03M	0.0293	9.85M	0.1099
EMKD(×2)	9.36M	0.0536	9.28M	0.0568	5.08M	0.1792
EMKD(×3)*	14.05M	0.0594	13.91M	0.0622	7.62M	0.1915

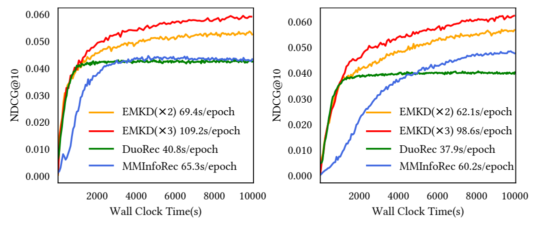
4.5 Adaptation to Other Models (RQ4)

EMKD trains an ensemble of bidirectional Transformers as sequence encoders, which can be viewed as an ensemble of BERT4Rec models. Since ensemble modeling and knowledge transfer is a model-agnostic approach, it can also be adapted to other sequence encoders. In this section, we apply EMKD to the ensemble of various base sequence encoders with a slight modification. Different from the Masked Language Modeling (MLM) style in bidirectional Transformers, other sequence encoders (e.g., GRU4Rec, Caser and SASRec) are autoregressive models that predict the next item based on previous items. Therefore, we need to switch the masked item prediction task to the next item prediction task. The attribute prediction task is also modified so that the model will predict the associate attributes at the next position.



(a) Beauty (b) ML-1M

Figure 4: Performance comparison (NDCG@10) of different models enhanced by EMKD on Beauty and ML-1M datasets. We design three variants for each group of base sequence encoder with 2,3,4 parallel networks respectively.



(a) Beauty (b) Toys

Figure 5: Training efficiency (NDCG@10) on Beauty and Toys datasets. The training speed of EMKD is slightly lower than MMInfoRec, while the convergence speed of EMKD is comparable with DuoRec.

Figure 4 presents the performances of different models enhanced by EMKD on Beauty and ML-1M datasets. We can see that all the base sequence encoders enhanced by our approach achieve better performance. This shows that ensemble modeling with contrastive knowledge distillation is a generalized approach that can be utilized to improve the performance of sequential recommenders.

4.6 Training Efficiency (RQ5)

A major concern for ensemble modeling is the training efficiency. To evaluate the training efficiency of our framework, we compare EMKD with two best-performed baselines: MMInfoRec and DuoRec. Training efficiency is evaluated from two aspects: training speed (average training time required for one epoch), and convergence speed (time required to achieve satisfactory performance). To visualize the performance, we save the checkpoint from every epoch and test them afterwards. All experiments use the default hyper-parameters and are carried out on a single Tesla V100 GPU.

Figure 5 presents the training efficiency of each model on Beauty and Toys datasets. We find out that EMKD with 2 parallel networks shows similar training speed (69.4s/epoch and 62.1s/epoch) with MMInfoRec (65.3s/epoch and 60.2s/epoch), while EMKD with 3 parallel networks (109.2s/epoch and 98.6s/epoch) is about 30% slower than MMInfoRec. Besides, DuoRec shows the best training speed (40.8s/epoch and 37.9s/epoch) on both datasets.

We also find out that EMKD almost converges as fast as DuoRec. The performance curve of EMKD coincides with the performance curve of DuoRec around the first 1000 seconds, indicating that EMKD and DuoRec show similar convergence speed during this interval. However, the performance of EMKD continues to improve afterwards, while DuoRec reaches its best performance. By comparison, MMInfoRec takes about 4000 seconds to converge to a satisfactory performance, which is slower than DuoRec and EMKD.

Compared with DuoRec and MMInfoRec, EMKD only brings a minor reduction in training efficiency. However, the performance improvement can be very significant compared with training a single network. Therefore, we think that it is worthwhile to adopt ensemble modeling methods in sequential recommendation.

4.7 Parameter Scaling (RQ6)

EMKD trains multiple bidirectional Transformer encoders as an ensemble network, which will increase the total number of parameters. To determine whether the good performance of EMKD results from the proposed knowledge transfer mechanisms or simply comes from the increase in the parameter size, we compare the performance of EMKD with other Transformer-based sequence encoders (SASRec and BERT4Rec) of similar parameter sizes. SASRec and BERT4Rec adopt two layers of Transformer blocks by default, and to scale the number of parameters, we stack more layers of Transformer blocks for SASRec and BERT4Rec so that they will have parameter sizes comparable to EMKD. Note that we count the parameters of both the embedding table and the sequence encoder(s) for each model, and we set the hidden dimensionality of each model as 256 for fair comparisons. Table 5 shows the parameter sizes and the performances of different models on three datasets. We can see that EMKD with 2 parallel networks has similar parameter size with a 6-layer SASRec or a 4-layer BERT4Rec, while EMKD with 3 parallel networks has similar parameter size with a 10-layer BERT4Rec. However, EMKD consistently outperforms SASRec and BERT4Rec of any parameter sizes, which verifies the effectiveness of the knowledge transfer mechanisms between the ensemble networks. We also find out that stacking more transformer blocks will not improve performance or even hurt performance in some cases, indicating that simply increasing the parameter size will not necessarily result in better performance for sequential recommendation.

5 CONCLUSION

In this paper, we present a novel framework called Ensemble Modeling with contrastive Knowledge Distillation for sequential recommendation (EMKD). Our framework adopts an ensemble of parallel networks to improve the overall prediction accuracy. To facilitate knowledge transfer between parallel networks, we propose a novel contrastive knowledge distillation approach that transfers knowledge from both the representation level and the logits level. We also design masked item prediction as the main task and attribute prediction as the auxiliary task for multi-task learning. Experiment results on three public benchmark datasets show that EMKD significantly outperforms baseline methods.

ACKNOWLEDGMENTS

This research is partially supported by the NSFC (61876117, 62176175), the major project of natural science research in Universities of Jiangsu Province (21KJA520004), Suzhou Science and Technology

Development Program (SYC2022139), the Priority Academic Program Development of Jiangsu Higher Education Institutions.

REFERENCES

[1] Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. 2021. Distilling Knowledge via Knowledge Review. In *CVPR*. 5008–5017.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*. 1597–1607.

[3] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. 2020. Improved Baselines with Momentum Contrastive Learning. In *arXiv preprint arXiv:2003.04297*.

[4] Xinlei Chen and Kaiming He. 2021. Exploring Simple Siamese Representation Learning. In *CVPR*. 15745–15753.

[5] Xinlei Chen, Saining Xie, and Kaiming He. 2021. An Empirical Study of Training Self-Supervised Vision Transformers. In *ICCV*. 9620–9629.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*. 4171–4186.

[7] Ziwei Fan, Zhiwei Liu, Yu Wang, Alice Wang, Zahra Nazari, Lei Zheng, Hao Peng, and Philip S. Yu. 2022. Sequential Recommendation via Stochastic Self-Attention. In *WWW*. 2036–2047.

[8] Zhiyuan Fang, Jianfeng Wang, Lijuan Wang, Lei Zhang, Yezhou Yang, and Zicheng Liu. 2021. SEED: Self-supervised Distillation for Visual Representation. In *ICLR*.

[9] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *EMNLP*. 6894–6910.

[10] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. 2018. Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. In *NeurIPS*. 8803–8812.

[11] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor W. Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS*. 8536–8546.

[12] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* (2015), 1–19.

[13] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *CVPR*. 9726–9735.

[14] Ruining He and Julian McAuley. 2016. Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation. In *ICDM*. 191–200.

[15] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. *ICLR*.

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531* (2015).

[17] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. 2017. Snapshot Ensembles: Train 1, Get M for Free. In *ICLR*.

[18] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*. 197–206.

[19] Kenji Kawaguchi. 2016. Deep Learning without Poor Local Minima. In *NeurIPS*. 586–594.

[20] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[21] Walid Krichene and Steffen Rendle. 2020. On Sampled Metrics for Item Recommendation. In *SIGKDD*. 1748–1757.

[22] Xu Lan, Xiatian Zhu, and Shaogang Gong. 2018. Knowledge Distillation by On-the-Fly Native Ensemble. In *NeurIPS*. 7528–7538.

[23] Julian McAuley, Christopher Targett, Javen Shi, and Anton van den Hengel. 2015. Image-based Recommendations on Styles and Substitutes. In *SIGIR*. 43–52.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*. 8024–8035.

[25] Ruihong Qiu, Zi Huang, and Hongzhi Yin. 2021. Memory Augmented Multi-Instance Contrastive Predictive Coding for Sequential Recommendation. In *ICDM*. 519–528.

[26] Ruihong Qiu, Zi Huang, Hongzhi Yin, and Zijian Wang. 2022. Contrastive Learning for Representation Degeneration Problem in Sequential Recommendation. In *WSDM*. 813–823.

[27] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *WWW*. 811–820.

[28] Guy Shani, Ronen I. Brafman, and David Heckerman. 2005. An MDP-Based Recommender System. *JMLR*, 1265–1295.

[29] Guocong Song and Wei Chai. 2018. Collaborative Learning for Deep Neural Networks. In *NeurIPS*. 1837–1846.

[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *JMLR* (2014), 1929–1958.

[31] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *CIKM*. 1441–1450.

[32] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *WSDM*. 565–573.

[33] Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2020. Contrastive Representation Distillation. In *ICLR*.

[34] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2018. Ensemble Adversarial Training: Attacks and Defenses. In *ICLR*.

[35] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748* (2018).

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS*. 6000–6010.

[37] Feng Wang and Huaping Liu. 2021. Understanding the Behaviour of Contrastive Loss. In *CVPR*. 2495–2504.

[38] Tongzhou Wang and Phillip Isola. 2020. Understanding Contrastive Representation Learning Through Alignment and Uniformity on the Hypersphere. In *ICML*. 9929–9939.

[39] Wei Wei, Chao Huang, Lianghao Xia, Yong Xu, Jiashu Zhao, and Dawei Yin. 2022. Contrastive Meta Learning with Behavior Multiplicity for Recommendation. In *WSDM*. 1120–1128.

[40] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive Learning for Sequential Recommendation. In *ICDE*. 1259–1273.

[41] Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. ConSERT: A Contrastive Framework for Self-Supervised Sentence Representation Transfer. In *ACL*. 5065–5075.

[42] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 2022. Are Graph Augmentations Necessary? Simple Graph Contrastive Learning for Recommendation. In *SIGIR*. 1294–1303.

[43] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. 2019. Feature-level Deeper Self-Attention Network for Sequential Recommendation. In *IJCAI*. 4320–4326.

[44] Ying Zhang, Tao Xiang, Timothy M. Hospedales, and Huchuan Lu. 2018. Deep Mutual Learning. In *CVPR*. 4320–4328.

[45] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-Rec: Self-Supervised Learning for Sequential Recommendation with Mutual Information Maximization. In *CIKM*. 1893–1902.