Warming Up Cold-Start CTR Prediction by Learning Item-Specific Feature Interactions

Yaqing Wang
Baidu Inc.
Baidu Research
Beijing, China
wangyaqing01@baidu.com

Hongming Piao
City University of Hong Kong
Department of Computer Science
Hong Kong SAR, China
hpiao6-c@my.cityu.edu.hk

Daxiang Dong
Baidu Inc.
Baidu AI Cloud
Beijing, China
dongdaxiang@baidu.com

Quanming Yao
Tsinghua University
Department of Electronic Engineering
Beijing, China
qyaoaa@tsinghua.edu.cn

Jingbo Zhou
Baidu Inc.
Baidu Research
Beijing, China
zhoujingbo@baidu.com

ABSTRACT

In recommendation systems, new items are continuously introduced, initially lacking interaction records but gradually accumulating them over time. Accurately predicting the click-through rate (CTR) for these items is crucial for enhancing both revenue and user experience. While existing methods focus on enhancing item ID embeddings for new items within general CTR models, they tend to adopt a global feature interaction approach, often overshadowing new items with sparse data by those with abundant interactions. Addressing this, our work introduces EmerG, a novel approach that warms up cold-start CTR prediction by learning item-specific feature interaction patterns. EmerG utilizes hypernetworks to generate an item-specific feature graph based on item characteristics, which is then processed by a Graph Neural Network (GNN). This GNN is specially tailored to provably capture feature interactions at any order through a customized message passing mechanism. We further design a meta learning strategy that optimizes parameters of hypernetworks and GNN across various item CTR prediction tasks, while only adjusting a minimal set of item-specific parameters within each task. This strategy effectively reduces the risk of overfitting when dealing with limited data. Extensive experiments on benchmark datasets validate that EmerG consistently performs the best given no, a few and sufficient instances of new items.

CCS CONCEPTS

• **Information systems → Recommender systems**; • **Computing methodologies → Supervised learning**; *Neural networks.*

KEYWORDS

Cold-start Recommendation, Warm Up, Click-through Rate Prediction, Few-Shot Learning, Hypernetworks, New Items

ACM Reference Format:
Yaqing Wang, Hongming Piao, Daxiang Dong, Quanming Yao, and Jingbo Zhou. 2024. Warming Up Cold-Start CTR Prediction by Learning Item-Specific Feature Interactions. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671784

1 INTRODUCTION

The cold-start problem presents a significant challenge in recommender systems [25], particularly evident as new items transition from having no user interactions (termed as cold-start phase) to accumulating a few initial clicks (termed as warm-up phase) in the industry landscape. Deep learning models, renowned for their capability to capture complex feature interactions, have shown promise in improving click-through rate (CTR) predictions, a critical metric for assessing the likelihood of user engagement with various items (e.g., movies, commodities, music) [1, 5, 9, 36]. However, these models typically rely on extensive datasets to achieve optimal performance, a requirement that poses a limitation in cold-start and warm-up phases. With their substantial parameter size, these models struggle to adapt efficiently to these phases characterized by limited interaction records, thereby exacerbating the challenge of making accurate CTR predictions and updating models without incurring significant costs.

Recent studies have focused on enhancing the initialization of item ID embeddings as a strategy to mitigate the item cold-start problem in recommender systems, which allows subsequent updates through gradient descent as interaction records become available in the warm-up phase [23, 24, 42, 45]. Then, they leverage general CTR backbones for further processing. However, they overlook a crucial aspect: the distinctiveness of feature interaction patterns across different users and items. This oversight limits the ability of these models to fully capture the nuanced dynamics of user-item interactions, potentially impacting the accuracy and effectiveness of CTR predictions in scenarios where personalized recommendations are crucial. For example, comparing high-priced luxury items with low-priced daily necessities reveals distinct interaction patterns. For high-priced luxury items, the interaction between the item's price and the user's income level is pivotal. Specifically, the second-order feature interaction <price, income> can be a determining factor in the user's willingness to purchase such items. As for low-priced daily necessities, the impact of the user's income level on purchasing decisions diminishes. In these instances, other feature interactions, such as those between the user's age and the item's category, become relatively more important. This variation underscores the necessity of modeling item-specific feature interaction patterns. While existing works all learn a global feature interaction pattern between users and items, which overwhelms new items with a limited number of interaction records by old items with abundant interaction records.

Recognizing the crucial role of feature interactions, we introduce EmerG to address CTR prediction of newly emerging items with incremental interaction data (from no interaction records to few and then abundant records) through the learning of item-specific feature graphs. Our contributions can be summarized as follows:

- We propose a unique method that emphasizes item-specific feature interactions, addressing the challenge of new item CTR prediction by reducing the overshadowing effect of older items with extensive data. Utilizing hypernetworks, we construct item-specific feature graphs with nodes as features and edges as their interactions, capturing complex interaction patterns unique to each item. We use a graph neural network (GNN) with a customized message passing process designed to provably capture feature interactions at any orders, which can be combined into nuanced and accurate predictions.
- To mitigate overfitting given limited data, we adopt a meta-learning strategy that optimizes parameters of hypernetworks and GNN across different item CTR prediction tasks with a few adjustments to item-specific parameters within each task. Besides, hypernetworks and GNN learned this way are expected to generalize to each task easily.
- We conduct extensive experiments on benchmark datasets and validate that EmerG performs the best for CTR prediction of emerging items. We also evaluate the performance given more training data, and find that EmerG consistently performs the best. Visualization of adjacency matrices which record item-specific feature graphs shows that EmerG can learn item-specific feature interactions properly.

2 RELATED WORKS

We briefly review four groups of methods relevant to CTR prediction of newly emerging items.

A. General CTR Models. General CTR models are applied universally without prioritizing underrepresented items. They mainly focus on modeling complex feature interactions, which is crucial for enhancing CTR prediction accuracy [3]. Historical advancements in this area show a progression from simple first-order interactions captured by linear models like logistic regression [27] to second-order interactions modeled by Factorization Machines (FM) [26], and to high-order interactions addressed by higher-order FMs (HOFMs) [2]. Various deep-learning models then automate the learning of these complex patterns. Both Wide&Deep [5] and DeepFM [9] employ hybrid architectures to handle second-order and higher interactions. DIN [43] dynamically captures user interests through an attention mechanism that adapts to varying ad features. AutoInt [28] introduces a multi-head self-attention mechanism [30] to model high-order feature interactions. LorentzFM [40] explores feature interactions in hyperbolic space to minimize parameter size. AFN [6] converts the power of each feature into the coefficient to be learned. FinalMLP [21] uses two multi-layer perceptron (MLP) networks in parallel, and equips them with feature selection and interaction aggregation layers. FINAL [44] introduces a factorized interaction layer for exponential growth in feature interaction learning. Considering that feature interactions can be conceptualized as graphs with nodes representing user and item features, graph neural network (GNN) [15] are used. Fi-GNN [17] learns to generate feature graphs where the edges are established according to the similarity between feature embeddings. FIVES [39] learns a global feature graph where edges are established by differentiable search from large-scale CTR datasets, thus new items with limited data can be underrepresented. GMT [22] models the interactions among items, users, and their features into a large heterogeneous graph, then feeds the local neighborhood of the target user-item pair for prediction. However, these general CTR models, designed for extensive datasets, often struggle during the cold-start & warm-up phases due to their substantial parameter size, which can lead to overfitting when adapted for new items. In contrast, EmerG introduces a specialized GNN that operates on item-specific feature graphs, generated via hypernetworks learned from diverse CTR tasks, ensuring precise modeling of feature interactions for new items with minimal parameter adjustments.

B. Methods for New Items without Interaction Records. Several methods specifically address the cold-start phase, where new items lack interaction records, while still maintaining model performance on older, established items. DropoutNet [31] trains neural networks with a dropout mechanism on input samples to infer missing data. Heater [46] employs a multi-gate mixture-of-experts approach to generate item embeddings. GAR [4] adopts an adversarial training strategy between a generator and a recommender to produce new item embeddings that mimic the distribution of old embeddings, deceiving the recommender systems. ALDI [12] learns to transfer the behavioral information of old items to new items. However, these methods do not consider the incorporation of incoming interaction records for new items and typically require re-training to accommodate the evolving interaction history of these items.

C. Methods for New Items with A Few Interaction Records. In scenarios where only a few instances of new items are available, which correspond to warm-up phases, few-shot learning [35] present a natural solution. Few-shot learning targets at generalizing to new tasks with a few labeled samples, which has been applied to image classification [8], query intent recognition [34] and drug discovery [33, 38, 41]. For CTR prediction, existing works typically approach the problem as a N-way K-shot task, where each of the N new items is associated with K labeled instances, then utilize the classic gradient-based meta-learning strategy [8]. MeLU [16] leverages this strategy to selectively adapt model parameters for new items through gradient descents. MAMO [7] enhances adaptation by incorporating an external memory mechanism. MetaHIN [20] utilizes heterogeneous information networks to exploit the rich semantic relationships between users and items. PAML [32] employs social relations to facilitate information sharing among similar users. More recent approaches have shifted from user-specific fine-tuning via gradient descent to amortization-based methods. These methods directly map user interaction histories to user-specific parameters, thus modulating the main network without iterative adjustments. TaNP [18] learns to modulate item-specific parameters based on item interaction records. ColdNAS [37] employs neural architecture search to optimize the modulation function and its application within the network. However, these methods struggle to handle new items that lack interaction records and cannot dynamically incorporate additional interaction records of new items as they become available.

D. Methods for Emerging Items with Incremental Interaction Records. To mirror the industry's dynamic evolution of new items, progressing from no interaction records to few and then abundant records, models are developed to manage these transitions smoothly. Existing efforts primarily enhance item ID embeddings for general CTR backbones. MetaE [24] employs gradient-based meta-learning to train an embedding generator. MWUF [45] transforms unstable item ID embeddings into stable ones using meta networks. CVAR [42] decodes new item ID embeddings from a distribution over item side information, circumventing additional data processing. GME [23] leverages information from neighboring old items for new item ID embedding generation. However, these methods generally optimize initial item ID embeddings while maintaining a global feature interaction pattern, thus failing to capture the unique characteristics and interaction dynamics of these items. In contrast, our EmerG addresses new item CTR prediction by tailoring feature interactions to each item with the help of hypernetworks. By learning with item-specific feature interactions, the risk of overwhelming new items by old items with abundant data is alleviated and prediction accuracy is enhanced.

3 PROBLEM FORMULATION

Let $\mathcal{V} = \{v_i\}$ denote a set of items where each item v_i is associated with N_v item features such as item ID, type and price. Similarly, let $\mathcal{U} = \{u_j\}$ denote a set of users where each user u_j is also associated with N_u user features such as user ID, age and hometown. When a user u_j clicks through an item v_i, label $y_{i,j} = 1$. Otherwise, $y_{i,j} = 0$.

During learning, the predictor is learned from a set of CTR prediction tasks $\mathcal{T}^{\text{old}} = \{\mathcal{T}_i\}_{t=1}^{N_t}$ sampled from old items, which can rapidly generalize to predict for tasks from new items that are unseen during training. Each task $\mathcal{T}_i$ corresponds to an old item v_i, with a training set $\mathcal{S}_i = \{(v_i, u_j, y_{i,j})\}_{j=1}^{N_s}$ containing existing interaction histories associated with v_i and a test set $\mathcal{Q}_i = \{(v_i, u_j, y_{i,j})\}_{j=1}^{N_q}$ containing interactions to predict whether u_j clicks through v_i. N_s and N_q are the number of interactions in $\mathcal{S}_i$ and $\mathcal{Q}_i$ respectively.

During testing, we consider CTR prediction for new items that start with no interaction records, then gradually gather a few, and eventually accumulate sufficient interaction records. Consider a task $\mathcal{T}_k$ associated with a new item v_k which is not considered during training, we handle three phases:

- Cold-start phase: No training set is provided.
- Warm-up phase: A training set $\mathcal{S}_k$ containing a few interaction records of v_k is given. There can be multiple warm-up phases where interaction records are gradually accumulated.
- Common phase: The training interaction records of v_k are accumulated to be sufficient.

For all three phases, the performance is evaluated on test set $\mathcal{Q}_k$.

4 THE PROPOSED EMERG

Aligning with the established understanding that feature interactions are crucial, we propose EmerG (Figure 1) to capture the uniqueness of items through their associated feature interaction patterns. We design two key components in EmerG: (ii) hypernetworks shared across different tasks to generate item-specific adjacent matrices encoding feature graphs; and (i) a GNN that operates on the generated item-specific feature graphs, whose message passing mechanism is specially tailored to provably capture feature interactions at any order. As we consider cold-start & warm-up phases, we further design a meta learning strategy that optimizes parameters of hypernetworks and GNN across various item CTR prediction tasks, while only adjusting a small set of item-specific parameters within each task. This strategy effectively reduces the risk of overfitting when dealing with limited data.

4.1 Embedding Layer

Given an instance (u, v), embedding layer maps the user features of u and item features of v into dense vectors. For the mth feature $f_m, m \in [1, N_v + N_u]$, its feature embedding $\mathbf{e}_m$ is obtained as

$$\mathbf{e}_m = \begin{cases} \boldsymbol{W}_{e,m} \cdot \text{one-hot}(f_m) & \text{if } f_m \text{ is single-valued} \\ \sum_e \boldsymbol{W}_{e,m} \cdot \text{multi-hot}(f_m) & \text{if } f_m \text{ is multi-valued} \\ \boldsymbol{W}_{e,m} \cdot f_m & \text{if } f_m \text{ is continuous} \end{cases}, \quad (1)$$

where $\boldsymbol{W}_{e,m}$ represents the embedding matrix corresponding to the mth feature, one-hot(f_m) represents the one-hot vector of single-valued feature f_m, and multi-hot(f_m) represents the multi-hot vector of multi-valued feature f_m.

4.2 Item-Specific Feature Graph Generation

We employ hypernetworks, following the strategy of Ha et al. [10], to generate item-specific feature graphs. Hypernetworks, small neural networks trained to generate parameters for a larger main network, present a unique challenge in their application, as their integration is highly problem-specific. In EmerG, hypernetworks are used to produce the initial adjacency matrix $\mathbf{A}_i^{(1)}$, encoding the item-specific feature graph for the first GNN layer. We streamline the process by allowing subsequent GNN layers to derive their adjacency matrices from the initial $\mathbf{A}_i^{(1)}$, optimizing storage efficiency without compromising the model's specificity to each item.

Consider task $\mathcal{T}_i$ for item v_i. For item features $f_1, \ldots, f_{N_v}$, item feature embeddings are denoted as $\mathbf{e}_{1,i}, \ldots, \mathbf{e}_{N_v,i}$ respectively. The feature graph [17, 39] is a graph where each node corresponds to a feature f_m, and the edge between two nodes records their interaction. We let our hypernetworks, which consists of $N_v + N_u$ subnetworks, produce a dense item-specific $\bar{\mathbf{A}}_i^{(1)} \in \mathbb{R}^{(N_v+N_u)\times(N_v+N_u)}$ which encodes the feature graph to be used in the first GNN layer.

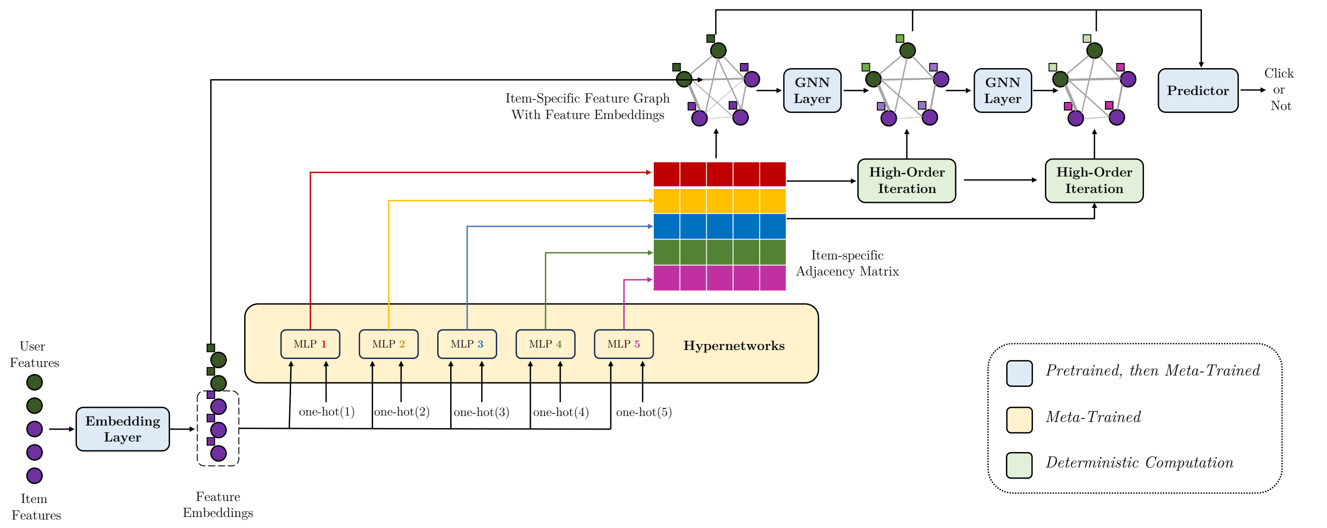


Figure 1: Illustration of the proposed EmerG, designed to enhance CTR predictions of newly emerging items through the learning of item-specific feature interaction patterns. EmerG uses hypernetworks to generate an initial item-specific adjacency matrix for a feature graph, with nodes representing user and item features and edges denoting their interactions, based on item feature embeddings. Higher-order adjacency matrices for subsequent GNN layers are generated from the initial matrix, reducing both model complexity and storage requirements. The GNN's message passing process is tailored to capture l-order feature interactions at the $l-1$th layer, enabling nuanced integration of various interaction orders for accurate predictions. EmerG optimizes the parameters of hypernetworks and GNN across diverse CTR prediction tasks to enhance generalization, while utilizing minimal item-specific parameters to capture the uniqueness of new items, which are adaptable with the introduction of additional item instances.

Denote the mth row of $\bar{\mathbf{A}}_i^{(1)}$ as $[\bar{\mathbf{A}}_i^{(1)}]_{m:}$, which is calculated as:

$$[\bar{\mathbf{A}}_i^{(1)}]_{m:} = \text{MLP}_{\mathbf{W}_a}([\mathbf{e}_{1,i}, \dots, \mathbf{e}_{N_v,i}, \text{one-hot}(m)]), \tag{2}$$

where $\text{MLP}_{\mathbf{W}_a}$ denotes a multi-layer perceptron (MLP) with parameter $\mathbf{W}_a$. Then, we generate $\bar{\mathbf{A}}_i^{(l)}$ as

$$\bar{\mathbf{A}}_i^{(l)} = \bar{\mathbf{A}}_i^{(l-1)} \cdot \bar{\mathbf{A}}_i^{(1)}. \tag{3}$$

This (3) returns $\bar{\mathbf{A}}_i^{(l)}$ as the matrix product of l copies of $\bar{\mathbf{A}}_i^{(1)}$. Therefore, $[\bar{\mathbf{A}}_i^{(l)}]_{mn}$ records the number of l-hop paths from node m to node n. In this way, we only need to keep one adjacency matrix (i.e., $\bar{\mathbf{A}}_i^{(1)}$) for each item v_i no matter how many GNN layers are used, which reduces parameter size.

Further, we take the following steps to refine adjacency matrices:

$$\hat{\mathbf{A}}_i^{(l)} = \text{sparsify}(\text{normalize}(\bar{\mathbf{A}}_i^{(l)}), K), \tag{4}$$

$$\tilde{\mathbf{A}}_i^{(l)} = ((\hat{\mathbf{A}}_i^{(l)})^\top + \hat{\mathbf{A}}_i^{(l)})/2, \tag{5}$$

$$\mathbf{A}_i^{(l)} = \text{normalize}(\text{mask}([\tilde{\mathbf{A}}_i^{(l-1)} \cdot \tilde{\mathbf{A}}_i^{(1)}], \tilde{\mathbf{A}}_i^{(l-1)})), \tag{6}$$

where normalize$(\cdot)$ applies min-max normalization to scale all the elements of a matrix to be in the range $[0, 1]$ and sets the diagonal elements of a matrix directly as 1, sparsify$(\cdot, K)$ keeps the top K largest elements and set the rest as 0, and mask$(\cdot, \tilde{\mathbf{A}}_i^{(l-1)})$ sets all zero elements of $\tilde{\mathbf{A}}_i^{(l-1)}$ in $\tilde{\mathbf{A}}_i^{(l)}$ as zero. From (4) to (6), we first sparsify the dense $\bar{\mathbf{A}}_i^{(l)}$ by (4) such that only two highly related features are connected. Further, because of the commutative law of $\odot$, we transform $\hat{\mathbf{A}}_i^{(l)}$ into a symmetric matrix by (5). Apart from the above-mentioned considerations, we expect that nodes disconnected in low-order feature graphs to be disconnected in high-order feature graphs. For example, if the message is not propagated from node n_2 to node n_1 in the lth GNN layer, the message of n_2 will be not propagated to n_1 in higher GNN layers. Thus, we apply (6) to obtain the final $\mathbf{A}_i^{(l)}$.

4.3 Customized Message Passing Process on Item-Specific Feature Graph

Upon the learned item-specific feature graphs, we use a GNN with a customized message passing process designed to provably capture feature interactions at any orders, which are then explicitly combined into the final CTR predictions.

We first describe the general mechanism of message passing. At the lth GNN layer, node embedding $\mathbf{h}_m^{(l)}$ of feature f_m is updated as

$$\mathbf{h}_m^{(l)} = \text{UPD}^{(l)}\left(\mathbf{h}_m^{(l-1)}, \text{AGG}^{(l)}\left(\left\{\mathbf{h}_n^{(l-1)} : f_n \in \mathcal{N}(f_m)\right\}\right)\right), \tag{7}$$

where $\text{UPD}^{(l)}(\cdot)$ updates node embedding of f_m as $\mathbf{h}_m^{(l)}$, $\text{AGG}^{(l)}(\cdot)$ aggregates node embeddings of neighbor nodes, $\mathcal{N}(f_m)$ contains neighbor nodes of node corresponding to feature f_m, and $\mathbf{h}_m^{(0)} = \mathbf{e}_m$. After N_l layers, node embedding $\mathbf{h}_m = \mathbf{h}_m^{(N_l)}$ is returned as the final feature representation.

In EmerG, we realize (7) as

$$\mathbf{h}_m^{(l)} = \mathbf{h}_m^{(l-1)} \odot \left[\sum\nolimits_{n=1}^{N_v+N_u} [\mathbf{A}_i^{(l-1)}]_{mn} \mathbf{W}_g^{(l-1)} \mathbf{h}_n^{(0)}\right], \tag{8}$$

where $\odot$ is the element-wise product, $\mathbf{A}_i^{(l-1)}$ is the final item-specific adjacency matrix obtained by (6), and $\mathbf{W}_g^{(l-1)}$ is a learnable parameter. Unlike existing GNNs [15, 17, 39] that aggregate $\mathbf{h}_m^{(l-1)}$ with $\mathbf{h}_n^{(l-1)}$, we aggregate $\mathbf{h}_m^{(l-1)}$ with $\mathbf{h}_n^{(0)}$. In this way, as Proposition 4.1 shows, the output of $(l-1)$th GNN layer is l-order feature interactions, which enables EmerG to explicitly model arbitrary-order feature interaction.

Proposition 4.1 (Efficacy of EmerG.). *With the customized message passing process defined in* (8), *the* $(l-1)$*th GNN layer captures* l*-order feature interactions.*

The proof is in Appendix A.1. One may consider integrating residual connections into GNN to model arbitrary-order feature interaction. However, as analyzed in Appendix A.2, incorporating residual connections will significantly elevate the maximum order of feature interaction.

With different orders of feature interactions, we then explicitly combine all nodes embeddings of each node f_m into the updated node embeddings $\hat{\mathbf{H}}_m$ of f_m by multi-head attention:

$$\begin{aligned}\text{attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) &= \text{softmax}(\mathbf{Q}\mathbf{K}^\top/\sqrt{N^d})\mathbf{V},\\ \text{head}_h &= \text{attention}(\mathbf{W}_{q,h}\mathbf{H}_m, \mathbf{W}_{k,h}\mathbf{H}_m, \mathbf{W}_{v,h}\mathbf{H}_m),\\ \hat{\mathbf{H}}_m &= [\text{head}_1;\ldots;\text{head}_{N_h}],\end{aligned}$$

where $\mathbf{H}_m = [\mathbf{h}_m^{(0)};\ldots;\mathbf{h}_m^{(N_l)}]$ contains N_l row vectors with length N_d, and N_h is the number of attention heads. Then, we estimate the contribution factor for each of the $N_v + N_u$ features as

$$[c_1,\ldots,c_{N_v+N_u}] = \text{sigmoid}\left(\text{MLP}_{\mathbf{W}_{c,1}}([\hat{\mathbf{H}}_1,\ldots,\hat{\mathbf{H}}_{N_v+N_u}])\right), \quad (9)$$

where $\text{MLP}_{\mathbf{W}_{c,1}}$ is parameterized by $\mathbf{W}_{c,1}$. Finally, we predict whether item v and user u interact as

$$\hat{y} = \sum\nolimits_{m=1}^{N_v+N_u} c_m \cdot \text{MLP}_{\mathbf{W}_{c,2}}(\hat{\mathbf{H}}_m), \quad (10)$$

where $\mathbf{W}_{c,2}$ is a trainable parameter.

4.4 Learning and Inference

To reduce the risk of overfitting when dealing with limited data, we introduce a meta learning strategy that optimizes parameters of hypernetworks and GNN across various item CTR prediction tasks, while only adjusting a minimal set of item-specific parameters within each task.

For simplicity, we denote hypernetworks as $\text{hyper}_{\theta_{\text{hyper}}}$ where $\theta_{\text{hyper}} = \mathbf{W}_a$ is the shared trainable parameter. Then, we denote the GNN as $\text{GNN}_{\theta_{\text{GNN}},\phi_i}$, where θ_{GNN} represents shared parameters including parameters of embedding layers $\{\mathbf{W}_{e,m}\}_{m=1}^{N_v+N_u}$, parameters of GNN layers $\{\mathbf{W}_g^{(l)}\}_{l=1}^{N_l}$, $\mathbf{W}_{q,h}$, $\mathbf{W}_{k,h}$, $\mathbf{W}_{v,h}$, parameters of predictor $\mathbf{W}_{c,1}, \mathbf{W}_{c,2}$, and ϕ_i represents item-specific parameters

$$\phi_i = \{\text{hyper}_{\theta_{\text{hyper}}}(v_i), \mathbf{e}_{\text{ID},i}\}, \quad (11)$$

which includes item-specific adjacency matrix $\mathbf{A}^{(1)}$ generated by hypernetworks and the randomized item ID embedding $\mathbf{e}_{\text{ID},i}$ of item v_i. We target at learning $\theta^*_{\text{GNN}}, \theta^*_{\text{hyper}}$, which can achieve good cold-start & warm-up performance on new item v_i by only generating ϕ_i and warming up ϕ_i with gradient descent.

We optimize EmerG w.r.t. the following objective calculated across N_t tasks from $\mathcal{T}^{\text{old}}$:

$$\sum\nolimits_i^{N_t} \gamma\mathcal{L}_{\mathcal{S}_i}(\theta_{\text{GNN}}, \phi_i) + (1-\gamma)\mathcal{L}_{\mathcal{Q}_i}(\theta_{\text{GNN}}, \phi_i'), \quad (12)$$

where γ is a hyperparameter to balance the contribution of two loss terms. In particular, the first term can represent the performance of cold-start phase [24] as the model has not been exposed to the labels in $\mathcal{S}_i$. We compute $\mathcal{L}_{\mathcal{S}_i}(\theta_{\text{GNN}}, \phi_i)$ as

$$\begin{aligned}&\mathcal{L}_{\mathcal{S}_i}(\theta_{\text{GNN}}, \phi_i) \quad (13)\\ &\equiv 1/|\mathcal{S}_i| \cdot \sum\nolimits_{(v_i,u_j,y_{ij})\in\mathcal{S}_i} \text{BCE}(y_{ij}, \text{GNN}_{\theta_{\text{GNN}},\phi_i}(v_i,u_j)),\end{aligned}$$

where $\text{BCE}(y,\hat{y}) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$ is the binary cross entropy. The second term in (12) represents the performance of warm-up phase after updating ϕ_i by a few new item instances provided in $\mathcal{S}_i$. We compute $\mathcal{L}_{\mathcal{Q}_i}(\theta_{\text{GNN}}, \phi_i')$ as

$$\begin{aligned}&\mathcal{L}_{\mathcal{Q}_i}(\theta_{\text{GNN}}, \phi_i') \quad (14)\\ &\equiv 1/|\mathcal{Q}_i| \cdot \sum\nolimits_{(v_i,u_j,y_{ij})\in\mathcal{Q}_i} \text{BCE}(y_{ij}, \text{GNN}_{\theta_{\text{GNN}},\phi_i'}(v_i,u_j)),\end{aligned}$$

with ϕ_i' obtained by performing gradient descent steps w.r.t (13):

$$\phi_i' = \phi_i - \alpha_1\nabla_{\phi_i}\mathcal{L}_{\mathcal{S}_i}(\theta_{\text{GNN}}, \phi_i), \quad (15)$$

where α_1 is the learning rate.

Algorithm 1 summarizes the training procedure of EmerG.

Algorithm 1 The training procedure of EmerG.

```
1:  randomly initialize $\theta_{\text{hyper}}$ and $\theta_{\text{GNN}}$;
2:  pretrain $\theta_{\text{GNN}}$ by old item instances;
3:  for $\mathcal{T}_i$ in $\mathcal{T}^{\text{old}}$ do
4:      sample $\mathcal{S}_i$ and $\mathcal{Q}_i$ for $\mathcal{T}_i$;
5:      randomly initialize item ID embedding $\mathbf{e}_{\text{ID},i}$ for $\mathcal{T}_i$;
6:      obtain feature embeddings $\mathbf{e}_{1,i},\ldots,\mathbf{e}_{N_v+N_u,i}$ by (1);
7:      generate $\bar{\mathbf{A}}_i^{(1)}$ of the first GNN layer by (2);
8:      get item-specific parameter $\phi_i = \{\bar{\mathbf{A}}_i^{(1)}, \mathbf{h}_{\text{ID},i}\}$;
9:      obtain $\mathbf{A}_i^{(l)}$ of subsequent GNN layers by (3)-(6);
10:     obtain feature representations $\mathbf{h}_{1,i},\ldots,\mathbf{h}_{N_v+N_u,i}$ by (8);
11:     obtain prediction $\hat{y}_{ij}$ by (10) for $(v_i,u_j,y_{i,j}) \in \mathcal{S}_i$;
12:     update $\phi_i$ as $\phi_i'$ by (15) with learning rate $\alpha_1$;
13:     optimize $\theta_{\text{hyper}}$ and $\theta_{\text{GNN}}$ w.r.t. (12) by gradient descents
        with learning rate $\alpha_2$;
14: end for
```

By learning from a set of tasks $\mathcal{T}^{\text{old}}$, the learned $\theta^*_{\text{GNN}}, \theta^*_{\text{hyper}}$ encode common knowledge. Consider task $\mathcal{T}_k$ of new item v_k. When v_k has no interaction record, namely $|\mathcal{S}_k| = 0$, we obtain its task-specific parameter ϕ_k and test its performance on the test set as cold-start phase performance. Given a few new item instances, we can update ϕ_k to take in the supervised information. Once ϕ_k is updated, we measure performance on the test set as warm-up phase performance. Algorithm 2 describes the testing procedure.

Algorithm 2 The testing procedure of EmerG.

```
1: optimized $\boldsymbol{\theta}^*_{\text{hyper}}$ and $\boldsymbol{\theta}^*_{\text{GNN}}$;
2: Consider task $\mathcal{T}_k$ of a new item $v_k$, given $\mathcal{S}_k$ with a few interaction records of $v_k$ and $\mathcal{Q}_k$ for evaluating CTR prediction performance of $v_k$;
3: if $|\mathcal{S}_k| = 0$ then
4:     randomly initialize item ID embedding $\mathbf{e}_{\text{ID},k}$ for $\mathcal{T}_k$;
5:     obtain feature embeddings $\mathbf{e}_{1,k}, \dots, \mathbf{e}_{N_v+N_u,k}$ by (1);
6:     generate $\bar{\mathbf{A}}^{(1)}_k$ of the first GNN layer by (2);
7:     get item-specific parameter $\boldsymbol{\phi}_k = \{\bar{\mathbf{A}}^{(1)}_k, \mathbf{e}_{\text{ID},k}\}$;
8:     obtain $\mathbf{A}^{(l)}_k$ of subsequent GNN layers by (3)-(6);
9:     obtain feature representations $\mathbf{h}_{1,k}, \dots, \mathbf{h}_{N_v+N_u,k}$ by (8);
10:    obtain prediction $\hat{y}_{kj}$ by (10) for $(v_k, u_j, y_{k,j}) \in \mathcal{S}_k$;
11:    measure performance on $\mathcal{Q}_k$;
12: else
13:    update $\mathbf{A}^{(l)}_k, l \in \{1 \cdots N_l\}$ by (3) to (6);
14:    update feature representations $\mathbf{h}_{1,k}, \dots, \mathbf{h}_{N_v+N_u,k}$ by (8);
15:    update $\boldsymbol{\phi}_k$ as $\boldsymbol{\phi}'_k$ by (15);
16:    obtain prediction $\hat{y}_{kj}$ by (10) for $(v_k, u_j, y_{k,j}) \in \mathcal{S}_k$;
17:    measure performance on $\mathcal{Q}_k$;
18: end if
```

5 EXPERIMENTS

5.1 Experimental Settings

Datasets. We use two benchmark datasets: (i) **MovieLens** [11]: a dataset containing 1 million interaction records on MovieLens, whose item features include movie ID, title, year of release, genres and user features include user ID, age, gender, occupation and zip-code; and (ii) **Taobao** [29]: a collection of 26 million ad click records on Taobao, whose item features include ad ID, position ID, category ID, campaign ID, advertiser ID, brand, price and user features include user ID, Micro group ID, cms_group_id, gender, age, consumption grade, shopping depth, occupation and city level. Following existing works [24, 42], we binarize the ratings of MovieLens, setting rating smaller than 4 as 0 and the others as 1.

Data Split. We adopt the public data split [24, 42, 45], group items according to their frequency: (i) **old items** which are items appearing in more than N interaction records, where $N = 200$ in MovieLens and $N = 2000$ in Taobao; and (ii) **new items** which are items appearing in less than N and larger than $3K$ interaction records, where K is set to 20 and 500 for MovieLens and Taobao respectively. To mimic the dynamic process where new items are gradually clicked by more users, the interaction records associated with new items are sorted by timestamp. We consider three successive warm-up phases, labeled as A, B, and C, each of which involves the introduction of a set of K new interaction records for each item. The rest interaction records form testing data for evaluation.

Experiment Pipeline. We adopt pipeline of existing works [24, 42, 45] to assess how a model adapts to new items over time. First, we use old item instances to pretrain the model, and directly evaluate the model performance on testing data of new items as cold-start phase performance. Then, we measure how model performs as it learns from a few training data in successive warm-up phases. In particular, we use the training data in warm-up phases A, B and C to sequentially update the model, and evaluate the performance of corresponding updated models on testing data.

Evaluation Metric. Following existing works [42], the performance is evaluated by (i) Area Under the Curve (AUC) [19] which represents the degree of separability, and (ii) F1 score [13] which is a harmonic mean of the precision and recall. Both AUC and F1 vary between 0 (worst) and 1 (best).

5.2 Performance Comparison

We compare the proposed EmerG[1] with the following four groups of baselines:

- A General CTR backbones pretrained using old item instances and fine-tuned by new item instances, including **DeepFM** [9], **Wide&Deep** [5], **AutoInt** [40], **AFN** [6], **Fi-GNN** [17], recent **FinalMLP** [21] and **FINAL** [44].
- B Methods for new items without interaction records, including **DropoutNet** [31] and **ALDI** [12].
- C Methods for new items with a few interaction records, including **MeLU** [16], **MAMO** [7], **TaNP** [18], and **ColdNAS** [37]. These methods cannot incorporate new item instances dynamically. Therefore, to accommodate the training interaction records provided in warm-up phases A, B, and C, we adopt a phased approach: initially, we use K interaction records from phase A as the support set to assess testing performance. Subsequently, we combine $2K$ records from phases A and B as the support set for a second evaluation. Finally, we incorporate $3K$ records from all three warm-up phases—A, B, and C—as the support set to conduct a third assessment of testing performance.
- D Methods for emerging items with incremental interaction records, which are the most relevant to ours. Existing works mainly equip general CTR backbones with the ability to generate and warm-up item ID embeddings for new items, including **MetaE** [24], **MWUF** [45], **GME** [23], and **CVAR** [42]. We use the classic DeepFM as the CTR backbone. Results of equipping these methods with other backbones are reported in Appendix C.1.

We implement the compared methods using public codes of the respective authors. More implementation details are provided in Appendix B.

Performance for Cold-Start & Warm-Up Phases. Table 1 shows the results. As shown, cold-start methods designed for cold-start & warm-up phases generally perform better. EmerG consistently performs the best in all four phases, validating the effectiveness of capturing item-specific feature interaction by hypernetworks. Few-shot methods for N-way K-shot settings obtains good performance in warm-up phase A. However, as the number of samples increases, it is unable to achieve greater performance improvement without complete retraining. General CTR backbones which are fine-tuned using the training sets perform worse, where FinalMLP performs the best. Recall that they randomly initialize item-specific parameters for new items, fine-tuning pretrained models by a small number of new item instances is not enough to obtain good performance. Particularly, note that the GNN-based CTR model Fi-GNN which uses item-user-specific feature interaction graphs perform

[1]Our code is available at https://github.com/LARS-group/EmerG.

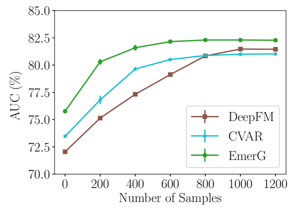
Table 1: Test performance obtained on MovieLens and Taobao. The best results are bolded, the second-best results are underlined.

MovieLens	Cold-Start Phase		Warm-Up Phase A		Warm-Up Phase B		Warm-Up Phase C	
	AUC(%)	F1(%)	AUC(%)	F1(%)	AUC(%)	F1(%)	AUC(%)	F1(%)
DeepFM	$71.48_{(0.15)}$	$60.96_{(0.22)}$	$75.21_{(0.27)}$	$64.09_{(0.01)}$	$77.70_{(0.26)}$	$66.18_{(0.15)}$	$79.45_{(0.19)}$	$67.81_{(0.14)}$
Wide&Deep	$69.44_{(0.29)}$	$59.53_{(0.31)}$	$74.82_{(0.29)}$	$64.44_{(0.21)}$	$77.58_{(0.25)}$	$66.82_{(0.24)}$	$79.09_{(0.22)}$	$67.67_{(0.27)}$
AutoInt	$68.64_{(0.24)}$	$59.63_{(0.14)}$	$75.60_{(0.31)}$	$64.93_{(0.36)}$	$77.65_{(0.33)}$	$66.84_{(0.42)}$	$79.20_{(0.34)}$	$67.77_{(0.36)}$
LorentzFM	$68.91_{(0.15)}$	$56.22_{(0.27)}$	$75.35_{(0.17)}$	$62.77_{(0.21)}$	$78.46_{(0.08)}$	$66.23_{(0.25)}$	$79.85_{(0.02)}$	$67.93_{(0.08)}$
AFN	$71.23_{(0.42)}$	$61.76_{(0.37)}$	$74.26_{(0.08)}$	$64.39_{(0.09)}$	$76.19_{(0.24)}$	$65.84_{(0.16)}$	$77.36_{(0.35)}$	$66.71_{(0.28)}$
Fi-GNN	$71.37_{(0.05)}$	$61.46_{(0.08)}$	$74.62_{(0.03)}$	$63.83_{(0.12)}$	$76.83_{(0.06)}$	$65.71_{(0.05)}$	$78.49_{(0.05)}$	$66.74_{(0.06)}$
FinalMLP	$69.51_{(0.06)}$	$60.59_{(0.17)}$	$78.48_{(0.12)}$	$67.34_{(0.10)}$	$78.47_{(0.16)}$	$67.27_{(0.07)}$	$79.07_{(0.17)}$	$68.00_{(0.10)}$
FINAL	$71.64_{(0.15)}$	$61.72_{(0.17)}$	$77.87_{(0.13)}$	$66.99_{(0.10)}$	$77.94_{(0.10)}$	$66.93_{(0.14)}$	$78.29_{(0.09)}$	$67.42_{(0.10)}$
DropoutNet	$72.94_{(0.17)}$	$62.43_{(0.18)}$	-	-	-	-	-	-
ALDI	$65.53_{(0.13)}$	$57.47_{(0.23)}$	-	-	-	-	-	-
MeLU	-	-	$77.54_{(0.06)}$	$66.71_{(0.11)}$	$79.43_{(0.10)}$	$68.51_{(0.05)}$	$80.26_{(0.03)}$	$68.13_{(0.06)}$
MAMO	-	-	$77.69_{(0.10)}$	$66.92_{(0.13)}$	$79.61_{(0.07)}$	$68.72_{(0.04)}$	$80.37_{(0.05)}$	$68.49_{(0.04)}$
TaNP	-	-	$79.15_{(0.10)}$	$\underline{68.39}_{(0.14)}$	$\underline{80.49}_{(0.17)}$	$\underline{69.43}_{(0.15)}$	$80.71_{(0.09)}$	$69.63_{(0.09)}$
ColdNAS	-	-	$77.45_{(0.03)}$	$67.01_{(0.03)}$	$77.88_{(0.12)}$	$67.25_{(0.21)}$	$78.06_{(0.09)}$	$67.31_{(0.11)}$
MetaE	$71.82_{(0.70)}$	$61.76_{(0.30)}$	$\underline{79.53}_{(0.25)}$	$67.96_{(0.15)}$	$80.27_{(0.09)}$	$68.31_{(0.12)}$	$80.47_{(0.04)}$	$68.46_{(0.12)}$
CVAR	$\underline{73.58}_{(0.21)}$	$63.15_{(0.12)}$	$78.23_{(0.10)}$	$67.03_{(0.26)}$	$80.28_{(0.06)}$	$68.76_{(0.12)}$	$\underline{81.06}_{(0.04)}$	$69.33_{(0.14)}$
GME	$71.54_{(0.13)}$	$\underline{64.31}_{(0.10)}$	$75.81_{(0.20)}$	$67.50_{(0.26)}$	$78.10_{(0.18)}$	$69.26_{(0.20)}$	$79.15_{(0.12)}$	$\underline{69.95}_{(0.16)}$
MWUF	$73.19_{(0.66)}$	$62.61_{(0.74)}$	$78.88_{(0.11)}$	$67.34_{(0.22)}$	$80.26_{(0.08)}$	$68.40_{(0.13)}$	$80.57_{(0.05)}$	$68.66_{(0.10)}$
EmerG	$\mathbf{75.44}_{(0.05)}$	$\mathbf{64.76}_{(0.15)}$	$\mathbf{79.92}_{(0.27)}$	$\mathbf{68.61}_{(0.24)}$	$\mathbf{81.28}_{(0.21)}$	$\mathbf{69.71}_{(0.14)}$	$\mathbf{81.82}_{(0.16)}$	$\mathbf{70.26}_{(0.14)}$

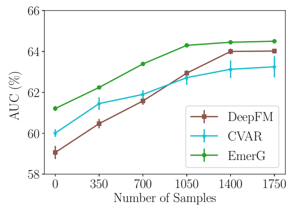
Taobao	Cold-Start Phase		Warm-Up Phase A		Warm-Up Phase B		Warm-Up Phase C	
	AUC(%)	F1(%)	AUC(%)	F1(%)	AUC(%)	F1(%)	AUC(%)	F1(%)
DeepFM	$59.01_{(0.84)}$	$13.47_{(0.42)}$	$60.68_{(0.65)}$	$14.27_{(0.20)}$	$61.51_{(0.64)}$	$14.56_{(0.34)}$	$62.34_{(0.54)}$	$15.00_{(0.24)}$
Wide&Deep	$59.07_{(0.44)}$	$13.65_{(0.06)}$	$60.92_{(0.56)}$	$14.25_{(0.10)}$	$61.69_{(0.51)}$	$14.56_{(0.15)}$	$62.33_{(0.46)}$	$14.75_{(0.13)}$
AutoInt	$55.69_{(1.37)}$	$12.14_{(0.22)}$	$58.65_{(1.26)}$	$13.61_{(0.51)}$	$59.43_{(1.20)}$	$13.84_{(0.36)}$	$60.07_{(1.13)}$	$14.19_{(0.39)}$
LorentzFM	$56.53_{(0.41)}$	$12.72_{(0.04)}$	$60.83_{(0.50)}$	$14.15_{(0.24)}$	$61.26_{(0.47)}$	$14.31_{(0.14)}$	$61.96_{(0.45)}$	$14.60_{(0.14)}$
AFN	$57.94_{(0.99)}$	$13.28_{(0.15)}$	$58.99_{(0.74)}$	$13.73_{(0.20)}$	$59.81_{(0.87)}$	$13.99_{(0.18)}$	$60.18_{(0.69)}$	$14.18_{(0.13)}$
Fi-GNN	$56.92_{(0.08)}$	$12.79_{(0.10)}$	$60.00_{(0.13)}$	$14.06_{(0.18)}$	$62.09_{(0.14)}$	$\underline{14.82}_{(0.14)}$	$62.46_{(0.21)}$	$14.90_{(0.05)}$
FinalMLP	$\underline{60.64}_{(0.12)}$	$13.57_{(0.04)}$	$\underline{63.44}_{(0.06)}$	$\underline{14.83}_{(0.03)}$	$\underline{63.49}_{(0.07)}$	$14.80_{(0.03)}$	$\underline{64.05}_{(0.02)}$	$15.05_{(0.03)}$
FINAL	$60.53_{(0.24)}$	$13.63_{(0.05)}$	$63.30_{(0.12)}$	$14.81_{(0.06)}$	$63.35_{(0.13)}$	$14.74_{(0.04)}$	$63.93_{(0.12)}$	$15.01_{(0.02)}$
DropoutNet	$60.41_{(0.09)}$	$13.53_{(0.02)}$	-	-	-	-	-	-
ALDI	$50.10_{(0.18)}$	$10.93_{(0.05)}$	-	-	-	-	-	-
MeLU	-	-	$61.37_{(0.17)}$	$14.09_{(0.17)}$	$62.48_{(0.04)}$	$14.34_{(0.05)}$	$63.07_{(0.07)}$	$14.64_{(0.11)}$
MAMO	-	-	$61.96_{(0.11)}$	$14.31_{(0.09)}$	$62.52_{(0.05)}$	$14.34_{(0.04)}$	$63.15_{(0.12)}$	$14.78_{(0.13)}$
TaNP	-	-	$55.67_{(0.22)}$	$11.92_{(0.31)}$	$55.85_{(0.16)}$	$12.07_{(0.16)}$	$56.19_{(0.09)}$	$12.08_{(0.11)}$
ColdNAS	-	-	$54.27_{(0.07)}$	$10.89_{(0.05)}$	$54.86_{(0.14)}$	$11.33_{(0.13)}$	$55.01_{(0.09)}$	$11.71_{(0.13)}$
MetaE	$59.75_{(0.37)}$	$13.58_{(0.06)}$	$61.19_{(0.26)}$	$14.01_{(0.09)}$	$62.06_{(0.31)}$	$14.41_{(0.10)}$	$62.87_{(0.32)}$	$14.71_{(0.07)}$
CVAR	$60.56_{(0.46)}$	$\underline{13.71}_{(0.13)}$	$62.54_{(0.19)}$	$14.38_{(0.06)}$	$63.17_{(0.10)}$	$14.69_{(0.05)}$	$63.95_{(0.18)}$	$\underline{15.09}_{(0.12)}$
GME	$60.57_{(0.23)}$	$13.32_{(0.33)}$	$62.55_{(0.17)}$	$13.96_{(0.22)}$	$63.29_{(0.05)}$	$14.39_{(0.12)}$	$63.85_{(0.13)}$	$14.52_{(0.08)}$
MWUF	$59.65_{(0.46)}$	$13.44_{(0.15)}$	$62.08_{(0.17)}$	$14.20_{(0.07)}$	$63.03_{(0.13)}$	$14.63_{(0.07)}$	$63.79_{(0.12)}$	$14.93_{(0.06)}$
EmerG	$\mathbf{61.58}_{(0.03)}$	$\mathbf{13.99}_{(0.05)}$	$\mathbf{63.56}_{(0.03)}$	$\mathbf{15.02}_{(0.06)}$	$\mathbf{63.76}_{(0.02)}$	$\mathbf{15.15}_{(0.01)}$	$\mathbf{64.22}_{(0.02)}$	$\mathbf{15.21}_{(0.02)}$

not well. This shows that too much freedom is not beneficial to capture feature interaction patterns under cold-start & warm-up phases. While in EmerG, we utilize hypernetworks to generate item-specific feature graphs, which is then processed by a GNN with customized message passing mechanism to capture arbitrary-order feature interaction, and optimize parameters by meta learning strategy. All these design considerations contributes the best performance obtained by EmerG. For computational overhead, EmerG is relatively more efficient in terms of both time and computational resources. See Appendix C.2 for a detailed comparison.

Performance Given Sufficient Training Samples. One might question how EmerG performs with an abundance of training samples for new items, referred to as the common phase, especially in comparison to traditional CTR backbones. Here, we set aside samples from original testing samples of new items (so the test set is smaller), use them to augment the experiment pipeline with more training samples, and evaluate the performance on the smaller test set. We compare EmerG with baselines which perform the best among CTR backbones and few-shot methods in Table 1. Figure 2 shows the testing AUC (%) with the number of training samples. Experimental results measured by testing F1 (%) are similar. As shown, all methods get better performance given more training samples. The classic CTR backbone DeepFM gradually outperforms CVAR which equips DeepFM with additional modules to generate item ID embeddings for new items. In contrast, EmerG consistently performs the best and converges to better performance than the others. This validates the effectiveness of our EmerG which can nicely capture item-specific feature interaction at different orders.



(a) MovieLens.



(b) Taobao.

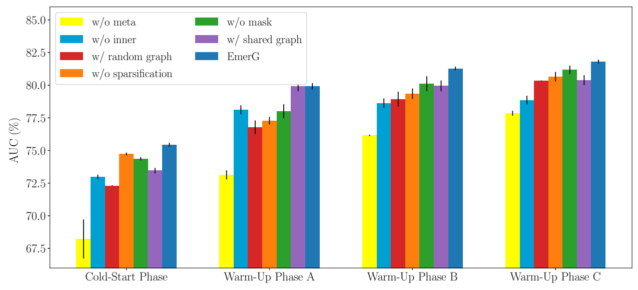
Figure 2: Comparing EmerG with DeepFM and CVAR given sufficient training samples.

5.3 Model Analysis

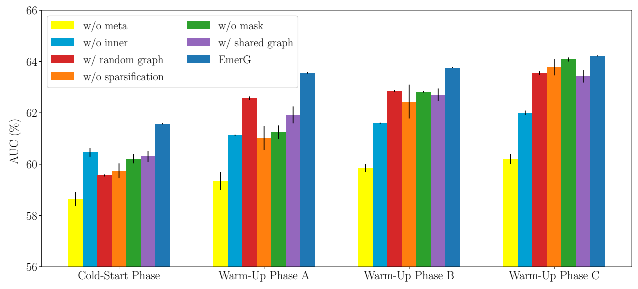
5.3.1 Ablation Study. We compare the proposed EmerG with the following variants: (i) **w/ random graph** generates the adjacency matrix $\mathbf{A}^{(1)}$ in (8) randomly; (ii) **w/o sparsification** does not apply (4) to sparsify the adjacency matrices; (iii) **w/o mask** does not apply (6) to enforce nodes which are disconnected in low-order feature graphs to be disconnected in higher-order feature graphs; (iv) **w/ shared graph** employs global shared adjacency matrices for all items, in contrast to EmerG, which utilizes item-specific adjacency matrices; (v) **w/o meta** learns both GNN and hypernetworks from old items, without forming tasks and employ a meta-learning strategy; and (vi) **w/o inner** directly uses $\boldsymbol{\phi}_i$ and does not update it to $\boldsymbol{\phi}'_i$ within each task.

Figure 3 shows the results. As shown, "w/ random graph" performs worse than EmerG which shows that the item-specific feature graphs generated by hypernetworks is meaningful. The performance gain of EmerG over "w/o sparsification" shows that a sparse feature graph where only closely-related nodes are connected can let the GNN model concentrate on useful messages. Comparing "w/o mask" to EmerG, the performance drop validates our assumption in (6). The mask operation also prevents the adjacency matrices from being too dense, which can be beneficial to prune unnecessary feature interactions and provide better explainability. We can also observe that"w/ shared graph" performs worse than EmerG. This validates that using global shared adjacency matrices cannot capture the various feature interaction patterns between different users and items. Finally, EmerG defeats "w/o meta" and "w/o inner", which underscores the necessity of both meta-learning across tasks and inner updates within each task.

5.3.2 Effect of Number of GNN Layers. As demonstrated in Proposition 4.1, we have customized the message passing process of the GNN to ensure that the lth layer encapsulates l-order feature interactions. Furthermore, we optimize this process by generating



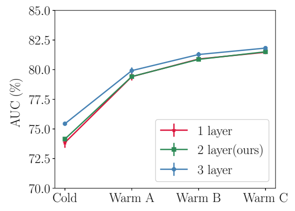
(a) MovieLens.



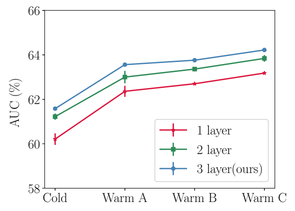
(b) Taobao.

Figure 3: Ablation study on MovieLens and Taobao.

adjacency matrices for subsequent GNN layers directly from the initial matrix provided by hypernetworks. This approach not only streamlines the architecture but also facilitates the extension to additional layers, thereby capturing higher-order feature interactions with ease. In this context, we investigate the influence of the number of GNN layers on performance across various datasets.



(a) MovieLens.

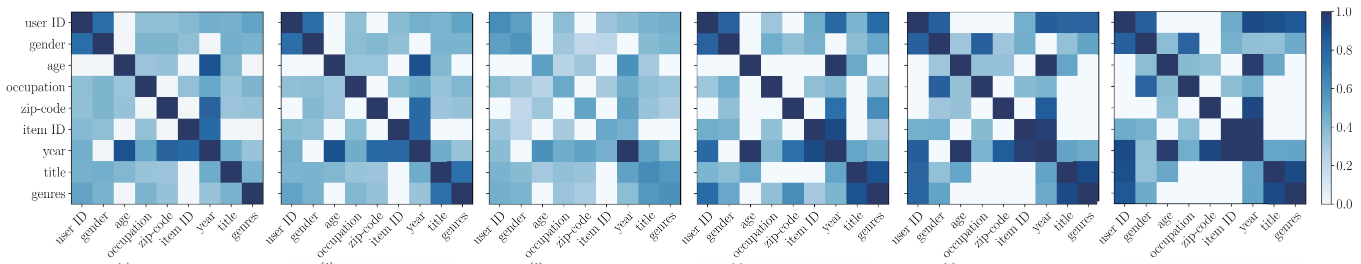


(b) Taobao.

Figure 4: Varying the number of GNN layers in EmerG.

Figure 4 shows the results. As can be seen, EmerG with different layer numbers obtain the best performance on different datasets: EmerG with 2 GNN layers performs the best on MovieLens while EmerG with 3 GNN layers achieves the best performance on Taobao. This shows that different datasets requires different number of GNN layers: larger datasets such as Taobao may need higher-order features than smaller ones such as MovieLens. By design, EmerG can easily meet this requirement.

5.3.3 Different Feature Interaction Functions. We consider using different feature interaction functions. Table 2 shows the results.



(a) $\mathbf{A}_{\mathrm{W}}^{(1)}$ in cold-start phase. (b) $\mathbf{A}_{\mathrm{L}}^{(1)}$ in cold-start phase. (c) $\mathbf{A}_{\mathrm{L}}^{(2)}$ in cold-start phase. (d) $\mathbf{A}_{\mathrm{L}}^{(1)}$ in warm-up phase A. (e) $\mathbf{A}_{\mathrm{L}}^{(1)}$ in warm-up phase B. (f) $\mathbf{A}_{\mathrm{L}}^{(1)}$ in warm-up phase C.

Figure 5: Visualizations of item-specific adjacency matrices of movie *Lawnmower Man 2: Beyond Cyberspace* ($\mathbf{A}_{\mathrm{W}}^{(i)}$) and movie *Waiting to Exhale* ($\mathbf{A}_{\mathrm{L}}^{(i)}$) in MovieLens generated by EmerG.

As shown, using element-wise product performs the best, which is adopted in EmerG.

Table 2: Test performance in AUC (%) obtained on MovieLens and Taobao. The best results are bolded.

MovieLens	Cold-Start	Warm-Up A	Warm-Up B	Warm-Up C
$\odot$	**75.44**	**79.92**	**81.28**	**81.82**
max	74.14	79.43	81.02	81.65
+	73.64	79.31	80.77	81.39

Taobao	Cold-Start	Warm-Up A	Warm-Up B	Warm-Up C
$\odot$	**61.58**	**63.56**	**63.76**	**64.22**
max	59.98	62.49	62.56	63.07
+	59.80	62.34	62.26	62.89

5.4 Case Study

Finally, we take movie *Lawnmower Man 2: Beyond Cyberspace* and movie *Waiting to Exhale* from MovieLens as new items, and visualize their adjacency matrices which record the item-specific feature graphs in Figure 5.

As can be seen, EmerG learns different task-specific feature graphs for different items. Comparing Figure 5(a) with Figure 5(b), we find that *Lawnmower Man 2: Beyond Cyberspace* has a particularly important second-order feature interaction ⟨genres, title⟩. The genre of *Lawnmower Man 2: Beyond Cyberspace* is science fiction while the genre of *Waiting to Exhale* is comedy. For a science fiction, its title often reflects its world view or theme, which is the key for people to judge whether they are interested. As for a comedy work, whether it is interesting or not is often irrelevant to the title.

Besides, EmerG can capture meaningful higher-order feature interactions. As shown, both ⟨year, age⟩ and ⟨year, zip-code⟩ are important second-order feature interactions, they contribute to discovering the third-order feature interaction ⟨year, age, zip-code⟩. In Figure 5(c), the relation between nodes of year, age and zip-code all become relatively important although <age, zip-code> is not important in Figure 5(b). It is easy to understand that the year of movies determines the age of people who are more likely to watch them. For example, elderly people generally prefer watching old movies. Apart from this, location which is indicated by zip-code also plays an important role: people in developed areas tend to be more receptive to new things. Therefore, area changes may lead to changes in the age of people who like the same movie, which validates the efficacy of the learned third-order feature interaction.

We can also observe that the item-specific feature graphs generated by our hypernetworks can roughly capture the feature interactions before seeing any training samples of new items. Although they are continuously optimized using training sets of warm-up phases, the changes are not sharp. As can be seen, the second-order feature interaction patterns are similar in Figure 5(b) and Figure 5(d) to Figure 5(f), with small changes to accommodate the training samples. Overall, we conclude that EmerG can learn reasonable adjacency matrices to capture item-specific feature interactions at different orders.

6 CONCLUSION

In this study, we underscore the critical role of feature interactions and introduce EmerG, a novel solution designed to capture the unique interaction patterns of items, effectively handling CTR prediction of newly emerging items with incremental interaction records. Our approach leverages hypernetworks to construct item-specific feature graphs, with nodes representing features and edges denoting their interactions, thus enabling the model to discern the intricate interaction patterns that characterize each item. We incorporate a graph neural network (GNN) equipped with a specialized message passing process, crafted to capture feature interactions across all orders, facilitating precise CTR predictions. To combat overfitting in scenarios with sparse data, we implement a meta-learning strategy that finely tunes parameters of hypernetworks and GNN across various item CTR prediction tasks, necessitating only minimal modifications to item-specific parameters for each task. Experimental results on real-world datasets show EmerG obtains the state-of-the-art performance on CTR prediction for new items that have no interaction history, a few interactions, or a substantial number of interactions. We expect this approach can be used to warm-up cold-start problems in other applications such as drug recommendation in the future.

ACKNOWLEDGMENT

We thank the anonymous reviewers for their valuable comments. This work is supported by National Key Research and Development Program of China under Grant 2023YFB2903904 and National Natural Science Foundation of China under Grant No. 92270106.

REFERENCES

[1] Jiang Bian, Jizhou Huang, Shilei Ji, Yuan Liao, Xuhong Li, Qingzhong Wang, Jingbo Zhou, Dejing Dou, Yaqing Wang, and Haoyi Xiong. 2023. Feynman: Federated Learning-based Advertising for Ecosystems-Oriented Mobile Apps Recommendation. *IEEE Transactions on Services Computing* 16, 5 (2023), 3361–3372.

[2] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. 2016. Higher-order factorization machines. In *Advances in Neural Information Processing Systems*. 3351–3359.

[3] Olivier Chapelle, Eren Manavoglu, and Romer Rosales. 2014. Simple and scalable response prediction for display advertising. *ACM Transactions on Intelligent Systems and Technology* 5, 4 (2014), 1–34.

[4] Hao Chen, Zefan Wang, Feiran Huang, Xiao Huang, Yue Xu, Yishi Lin, Peng He, and Zhoujun Li. 2022. Generative adversarial framework for cold-start item recommendation. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2565–2571.

[5] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Workshop on Deep Learning for Recommender Systems*. 7–10.

[6] Weiyu Cheng, Yanyan Shen, and Linpeng Huang. 2020. Adaptive factorization network: Learning adaptive-order feature interactions. In *AAAI Conference on Artificial Intelligence*. 3609–3616.

[7] Manqing Dong, Feng Yuan, Lina Yao, Xiwei Xu, and Liming Zhu. 2020. MAMO: Memory-augmented meta-optimization for cold-start recommendation. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 688–697.

[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*. 1126–1135.

[9] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A factorization-machine based neural network for CTR prediction. In *International Joint Conference on Artificial Intelligence*. 1725–1731.

[10] David Ha, Andrew Dai, and Quoc V Le. 2017. Hypernetworks. In *International Conference on Learning Representations*.

[11] F Maxwell Harper and Joseph A Konstan. 2015. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2015), 1–19.

[12] Feiran Huang, Zefan Wang, Xiao Huang, Yufeng Qian, Zhetao Li, and Hao Chen. 2023. Aligning distillation for cold-start item recommendation. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1147–1157.

[13] Hao Huang, Haihua Xu, Xianhui Wang, and Wushour Silamu. 2015. Maximum F1-score discriminative training criterion for automatic mispronunciation detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23, 4 (2015), 787–797.

[14] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

[15] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

[16] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. MeLU: Meta-learned user preference estimator for cold-start recommendation. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1073–1082.

[17] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-GNN: Modeling feature interactions via graph neural networks for CTR prediction. In *ACM International Conference on Information and Knowledge Management*. 539–548.

[18] Xixun Lin, Jia Wu, Chuan Zhou, Shirui Pan, Yanan Cao, and Bin Wang. 2021. Task-adaptive neural process for user cold-start recommendation. In *The Web Conference*. 1306–1316.

[19] Charles X Ling, Jin Huang, Harry Zhang, et al. 2003. AUC: A statistically consistent and more discriminating measure than accuracy. In *International Joint Conference on Artificial Intelligence*. 519–524.

[20] Yuanfu Lu, Yuan Fang, and Chuan Shi. 2020. Meta-learning on heterogeneous information networks for cold-start recommendation. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1563–1573.

[21] Kelong Mao, Jieming Zhu, Liangcai Su, Guohao Cai, Yuru Li, and Zhenhua Dong. 2023. FinalMLP: An enhanced two-stream MLP model for CTR prediction. In *AAAI Conference on Artificial Intelligence*. 4552–4560.

[22] Erxue Min, Yu Rong, Tingyang Xu, Yatao Bian, Da Luo, Kangyi Lin, Junzhou Huang, Sophia Ananiadou, and Peilin Zhao. 2022. Neighbour interaction based click-through rate prediction via graph-masked transformer. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. 353–362.

[23] Wentao Ouyang, Xiuwu Zhang, Shukui Ren, Li Li, Kun Zhang, Jinmei Luo, Zhaojie Liu, and Yanlong Du. 2021. Learning graph meta embeddings for cold-start ads in click-through rate prediction. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1157–1166.

[24] Feiyang Pan, Shuokai Li, Xiang Ao, Pingzhong Tang, and Qing He. 2019. Warm up cold-start advertisements: Improving CTR predictions via learning to learn ID embeddings. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. 695–704.

[25] Yoon-Joo Park and Alexander Tuzhilin. 2008. The long tail of recommender systems and how to leverage it. In *ACM Conference on Recommender Systems*. 11–18.

[26] Steffen Rendle. 2010. Factorization machines. In *IEEE International Conference on Data Mining*. 995–1000.

[27] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: Estimating the click-through rate for new ads. In *International Conference on World Wide Web*. 521–530.

[28] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. AutoInt: Automatic feature interaction learning via self-attentive neural networks. In *ACM International Conference on Information and Knowledge Management*. 1161–1170.

[29] Tianchi. 2018. Taobao Display Ads Click Dataset. https://tianchi.aliyun.com/dataset/dataDetail?dataId=56.

[30] 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 *Advances in Neural Information Processing Systems*. 5998–6008.

[31] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. DropoutNet: Addressing cold start in recommender systems. In *Advances in Neural Information Processing Systems*. 4957–4966.

[32] Li Wang, Binbin Jin, Zhenya Huang, Hongke Zhao, Defu Lian, Qi Liu, and Enhong Chen. 2021. Preference-adaptive meta-learning for cold-start recommendation.. In *International Joint Conference on Artificial Intelligence*. 1607–1614.

[33] Yaqing Wang, Abulikemu Abuduweili, Quanming Yao, and Dejing Dou. 2021. Property-aware relation networks for few-shot molecular property prediction. In *Advances in Neural Information Processing Systems*. 17441–17454.

[34] Yaqing Wang, Song Wang, Yanyan Li, and Dejing Dou. 2022. Recognizing medical search query intent by few-shot learning. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. 502–512.

[35] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *Comput. Surveys* 53, 3 (2020), 1–34.

[36] Yan Wen, Chen Gao, Lingling Yi, Liwei Qiu, Yaqing Wang, and Yong Li. 2023. Efficient and Joint Hyperparameter and Architecture Search for Collaborative Filtering. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2547–2558.

[37] Shiguang Wu, Yaqing Wang, Qinghe Jing, Daxiang Dong, Dejing Dou, and Quanming Yao. 2023. ColdNAS: Search to modulate for user cold-start recommendation. In *The Web Conference*. 1021–1031.

[38] Shiguang Wu, Yaqing Wang, and Quanming Yao. 2024. PACIA: Parameter-efficient adapter for few-shot molecular property prediction. In *International Joint Conference on Artificial Intelligence*.

[39] Yuexiang Xie, Zhen Wang, Yaliang Li, Bolin Ding, Nezihe Merve Gürel, Ce Zhang, Minlie Huang, Wei Lin, and Jingren Zhou. 2021. FIVES: Feature interaction via edge search for large-scale tabular data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3795–3805.

[40] Canran Xu and Ming Wu. 2020. Learning feature interactions with lorentzian factorization machine. In *AAAI Conference on Artificial Intelligence*. 6470–6477.

[41] Quanming Yao, Zhenqian Shen, Yaqing Wang, and Dejing Dou. 2024. Property-aware relation networks for few-shot molecular property prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[42] Xu Zhao, Yi Ren, Ying Du, Shenzheng Zhang, and Nian Wang. 2022. Improving item cold-start recommendation via model-agnostic conditional variational autoencoder. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2595–2600.

[43] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1059–1068.

[44] Jieming Zhu, Qinglin Jia, Guohao Cai, Quanyu Dai, Jingjie Li, Zhenhua Dong, Ruiming Tang, and Rui Zhang. 2023. FINAL: Factorized interaction layer for CTR prediction. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2006–2010.

[45] Yongchun Zhu, Ruobing Xie, Fuzhen Zhuang, Kaikai Ge, Ying Sun, Xu Zhang, Leyu Lin, and Juan Cao. 2021. Learning to warm up cold item embeddings for cold-start recommendation with meta scaling and shifting networks. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1167–1176.

[46] Ziwei Zhu, Shahin Sefati, Parsa Saadatpanah, and James Caverlee. 2020. Recommendation for new users and new items via randomized training and mixture-of-experts transformation. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1121–1130.

A MESSAGE PASSING MECHANISM

A.1 Proof of Proposition 4.1

PROOF. In (8), node embedding $\mathbf{h}_m^{(l-1)}$ aggregates from multiple first-order node embeddings $\mathbf{h}_n^{(0)}$ to generate $\mathbf{h}_m^{(l)}$. Therefore,

$$\text{order}(\mathbf{h}_m^{(0)}) = 1,$$
$$\text{order}(\mathbf{h}_m^{(l)}) = \text{order}(\mathbf{h}_m^{(l-1)}) + 1,$$

where $\text{order}(\mathbf{h}_m^{(l)})$ represents the order of $\mathbf{h}_m^{(l)}$. We can conclude

$$\text{order}(\mathbf{h}_m^{(l)}) = l + 1.$$

After merging all nodes embedding with multi-head self-attention, all orders of features we can obtain after $l-1$ GNN layers are

$$\text{order}(\hat{\mathbf{H}}_m) = \{1, 2, ..., l, l+1\}.$$

□

A.2 Comparing with GNN with Residual Connections

Although GNN with residual connections can model arbitrary-order feature interaction [17], the maximum order of feature interaction will increase drastically. To see this, instead of using (8), (7) can be realized with residual connection as

$$\mathbf{h}_m^{(l)} = \mathbf{h}_m^{(l-1)} + \mathbf{h}_m^{(l-1)} \odot \sum\nolimits_{n=1}^{N_v+N_u} [\mathbf{A}_i^{(l-1)}]_{mn} \mathbf{W}_g^{(l-1)} \mathbf{h}_n^{(l-1)}.$$

As $\mathbf{h}_m^{(l-1)}$ and $\mathbf{h}_n^{(l-1)}$ have the same maximum order, when we aggregate them via $\odot$, we have:

$$\text{maxorder}(\mathbf{h}_m^{(l)}) = \text{maxorder}(\mathbf{h}_m^{(l-1)}) \times 2.$$

In other words, using residue connections will lead to too many noisy high-order feature interactions. Consequently, it is also challenging to determine which feature interactions contribute to the prediction, resulting in low interpretability.

B IMPLEMENTATION DETAILS

All results are averaged over five runs and are obtained on a 32GB NVIDIA Tesla V100 GPU. We use Adam optimizer [14]. To search for the appropriate hyperparameters, we set aside 20% old items and form validation set using their samples. The performance is then directly evaluated on the validation set of these items, which corresponds to cold-start phase. When the hyperparameters are found by grid search, we put back samples of these old items, then follow the experiment pipeline described in Section 5.1 and report the results. The hyperparameters and their range used by EmerG are summarized in Table 3.

C MORE EXPERIMENTAL RESULTS

C.1 Comparing with Existing Methods Equipped with Different Backbones

In Section 5.2, we employ DeepFM as the backbone for methods in Group D. Despite this, results in Table 1 indicate that FinalMLP generally surpasses DeepFM, particularly in the warm-up phases, though not in the cold-start phases. Therefore, we further integrate FinalMLP, the top-performing backbone from Group A, into methods in Group D. Results are reported in Table 4. Notably, FinalMLP, when utilized as a backbone for cold-start methods, does not exceed the performance of configurations using DeepFM. This suggests that more recent CTR backbones cannot effectively handle the CTR prediction of newly emerging items.

Table 3: Hyperparameters used by EmerG. $N' = N_v + N_u$.

Hyperparameter	Range	MovieLens	Taobao
number of GNN layers	$[1, 2, 3]$	2	3
K in (4)	$[0, 1, \cdots, N' \cdot N'$	$\frac{N' \cdot N'}{2}$	$\frac{N' \cdot N'}{2}$
γ in loss function	$[1e-2, \ldots, 1]$	0.1	0.1
number of heads	$[1, 2, \ldots, 5]$	3	3
embedding dimension	$[10, 11, \ldots, 20]$	16	16
batch size	$[64, 128, \cdots, 1024]$	512	512
pretraining learning rate	$[1e-4, 1e-1]$	0.005	0.001
pretraining epochs	$[1, 2, \ldots, 20]$	2	1
meta-training learning rate α_2	$[1e-4, 1e-1]$	0.001	0.0001
meta-training epochs	$[1, 2, \ldots, 20]$	11	3
update learning rate α_1 during meta-training	$[1e-4, 1e-1]$	0.01	0.001
update learning rate α_1 during warming-up	$[1e-4, 1e-1]$	0.01	0.01
warming-up epochs	$[1, 2, \ldots, 20]$	11	16

C.2 Computational Overhead

Table 5 shows a detailed comparison of the computational overhead for all compared methods. As indicated, EmerG demonstrates relatively lower time and space requirements.

Table 5: Computational overhead of compared methods on MovieLens. Time is reported as seconds per epoch.

		Training Time	Test Time	# Para.
DeepFM		831.30	43.59	0.51
Wide&Deep		725.15	45.30	0.51
AutoInt		877.75	48.90	0.54
LorentzFM		922.75	52.21	0.51
AFN		926.65	46.30	10.29
Fi-GNN		970.11	56.11	0.53
FinalMLP		1013.55	53.30	2.30
FINAL		944.84	49.00	1.03
MeLU		1123.94	52.11	0.51
MAMO		1299.84	52.71	0.71
TaNP		1089.23	31.25	0.54
ColdNAS		1190.11	25.81	1.81
ALDI		576.40	16.40	0.51
DropoutNet	DeepFM	827.20	43.20	0.51
	FinalMLP	1043.23	54.26	2.3
MetaE	DeepFM	1235.44	44.27	0.51
	FinalMLP	1319.18	52.70	2.30
CVAR	DeepFM	2372.70	44.20	0.52
	FinalMLP	2516.50	52.89	2.31
GME	DeepFM	1099.30	43.99	0.52
	FinalMLP	1101.44	54.21	2.31
MWUF	DeepFM	1784.95	43.67	0.52
	FinalMLP	2012.60	53.44	2.31
EmerG		996.26	46.10	0.82

Table 4: Comparing EmerG with methods for emerging items with incremental interaction records, using various backbones. Test performance obtained on MovieLens and Taobao. The best results are bolded, the second-best results are underlined.

MovieLens		Cold-start Phase		Warm-up Phase A		Warm-up Phase B		Warm-up Phase C	
		AUC(%)	F1(%)	AUC(%)	F1(%)	AUC(%)	F1(%)	AUC(%)	F1(%)
DropoutNet	DeepFM	$72.94_{(0.17)}$	$62.43_{(0.18)}$	$78.69_{(0.01)}$	$67.17_{(0.05)}$	$78.73_{(0.06)}$	$67.12_{(0.04)}$	$79.28_{(0.05)}$	$67.76_{(0.09)}$
	FinalMLP	$72.78_{(0.10)}$	$62.41_{(0.16)}$	$78.55_{(0.13)}$	$67.39_{(0.11)}$	$78.46_{(0.07)}$	$67.18_{(0.00)}$	$78.96_{(0.07)}$	$67.78_{(0.10)}$
MetaE	DeepFM	$71.82_{(0.70)}$	$61.76_{(0.30)}$	$\underline{79.53}_{(0.25)}$	$67.96_{(0.15)}$	$80.27_{(0.09)}$	$68.31_{(0.12)}$	$80.47_{(0.04)}$	$68.46_{(0.12)}$
	FinalMLP	$59.50_{(4.80)}$	$53.22_{(4.77)}$	$72.38_{(3.44)}$	$62.11_{(3.03)}$	$75.34_{(3.64)}$	$64.35_{(3.19)}$	$76.98_{(3.05)}$	$65.83_{(2.64)}$
CVAR	DeepFM	$\underline{73.58}_{(0.21)}$	$63.15_{(0.12)}$	$78.23_{(0.10)}$	$67.03_{(0.26)}$	$80.28_{(0.06)}$	$68.76_{(0.12)}$	$\underline{81.06}_{(0.04)}$	$69.33_{(0.14)}$
	FinalMLP	$65.91_{(2.58)}$	$59.02_{(1.14)}$	$77.33_{(0.16)}$	$65.94_{(0.45)}$	$77.90_{(0.37)}$	$66.58_{(0.26)}$	$78.86_{(0.32)}$	$67.62_{(0.17)}$
GME	DeepFM	$71.54_{(0.13)}$	$64.31_{(0.10)}$	$75.81_{(0.20)}$	$67.50_{(0.26)}$	$78.10_{(0.18)}$	$69.26_{(0.20)}$	$79.15_{(0.12)}$	$\underline{69.95}_{(0.16)}$
	FinalMLP	$71.56_{(0.28)}$	$63.79_{(0.37)}$	$76.48_{(0.36)}$	$67.81_{(0.44)}$	$78.94_{(0.28)}$	$68.86_{(0.34)}$	$80.04_{(0.19)}$	$69.79_{(0.34)}$
MWUF	DeepFM	$73.19_{(0.66)}$	$62.61_{(0.74)}$	$78.88_{(0.11)}$	$67.34_{(0.22)}$	$80.26_{(0.08)}$	$68.40_{(0.13)}$	$80.57_{(0.05)}$	$68.66_{(0.10)}$
	FinalMLP	$69.02_{(0.41)}$	$59.56_{(0.41)}$	$78.06_{(0.37)}$	$66.88_{(0.39)}$	$79.58_{(0.15)}$	$68.23_{(0.07)}$	$80.12_{(0.10)}$	$68.69_{(0.08)}$
EmerG		$\mathbf{75.44}_{(0.05)}$	$\mathbf{64.76}_{(0.15)}$	$\mathbf{79.92}_{(0.27)}$	$\mathbf{68.61}_{(0.24)}$	$\mathbf{81.28}_{(0.21)}$	$\mathbf{69.71}_{(0.14)}$	$\mathbf{81.82}_{(0.16)}$	$\mathbf{70.26}_{(0.14)}$

Taobao		Cold-start Phase		Warm-up Phase A		Warm-up Phase B		Warm-up Phase C	
		AUC(%)	F1(%)	AUC(%)	F1(%)	AUC(%)	F1(%)	AUC(%)	F1(%)
DropoutNet	DeepFM	$60.41_{(0.09)}$	$13.53_{(0.02)}$	$62.48_{(0.26)}$	$14.55_{(0.10)}$	$62.60_{(0.26)}$	$14.68_{(0.12)}$	$63.12_{(0.17)}$	$14.82_{(0.08)}$
	FinalMLP	$\underline{60.86}_{(0.14)}$	$13.69_{(0.09)}$	$\underline{63.37}_{(0.08)}$	$14.75_{(0.03)}$	$\underline{63.43}_{(0.01)}$	$\underline{14.84}_{(0.03)}$	$\underline{63.98}_{(0.03)}$	$15.04_{(0.04)}$
MetaE	DeepFM	$59.75_{(0.37)}$	$13.58_{(0.06)}$	$61.19_{(0.26)}$	$14.01_{(0.09)}$	$62.06_{(0.31)}$	$14.41_{(0.10)}$	$62.87_{(0.32)}$	$14.71_{(0.07)}$
	FinalMLP	$59.71_{(1.00)}$	$13.12_{(0.91)}$	$62.58_{(0.45)}$	$14.87_{(0.13)}$	$62.68_{(0.43)}$	$14.79_{(0.12)}$	$63.30_{(0.44)}$	$15.13_{(0.08)}$
CVAR	DeepFM	$60.56_{(0.46)}$	$\underline{13.71}_{(0.13)}$	$62.54_{(0.19)}$	$14.38_{(0.06)}$	$63.17_{(0.10)}$	$14.69_{(0.05)}$	$63.95_{(0.18)}$	$15.09_{(0.12)}$
	FinalMLP	$60.55_{(0.49)}$	$13.83_{(0.23)}$	$63.24_{(0.47)}$	$\underline{14.99}_{(0.22)}$	$63.25_{(1.18)}$	$15.03_{(0.50)}$	$63.79_{(0.77)}$	$\underline{15.15}_{(0.43)}$
GME	DeepFM	$60.57_{(0.23)}$	$13.32_{(0.33)}$	$62.55_{(0.17)}$	$13.96_{(0.22)}$	$63.29_{(0.05)}$	$14.39_{(0.12)}$	$63.85_{(0.13)}$	$14.52_{(0.08)}$
	FinalMLP	$60.78_{(0.15)}$	$13.76_{(0.06)}$	$63.10_{(0.18)}$	$14.89_{(0.16)}$	$63.12_{(0.04)}$	$14.81_{(0.07)}$	$63.76_{(0.19)}$	$14.96_{(0.14)}$
MWUF	DeepFM	$59.65_{(0.46)}$	$13.44_{(0.15)}$	$62.08_{(0.17)}$	$14.20_{(0.07)}$	$63.03_{(0.13)}$	$14.63_{(0.07)}$	$63.79_{(0.12)}$	$14.93_{(0.06)}$
	FinalMLP	$60.36_{(0.11)}$	$13.73_{(0.08)}$	$63.26_{(0.07)}$	$14.00_{(0.02)}$	$63.37_{(0.22)}$	$14.79_{(0.05)}$	$63.96_{(0.23)}$	$15.09_{(0.05)}$
EmerG		$\mathbf{61.58}_{(0.03)}$	$\mathbf{13.99}_{(0.05)}$	$\mathbf{63.56}_{(0.03)}$	$\mathbf{15.02}_{(0.06)}$	$\mathbf{63.76}_{(0.02)}$	$\mathbf{15.15}_{(0.01)}$	$\mathbf{64.22}_{(0.02)}$	$\mathbf{15.21}_{(0.02)}$