HyperFormer: Learning Expressive Sparse Feature Representations via Hypergraph Transformer

Kaize Ding*
Arizona State University
kaize.ding@asu.edu

Albert Jiongqian Liang
Google Research
jiongqian@google.com

Bryan Perrozi
Google Research
hubris@google.com

Ting Chen
Google DeepMind
iamtingchen@google.com

Ruoxi Wang
Google DeepMind
ruoxi@google.com

Lichan Hong
Google DeepMind
lichan@goole.com

Ed H. Chi
Google DeepMind
edchi@google.com

Huan Liu
Arizona State University
huanliu@asu.edu

Derek Zhiyuan Cheng
Google DeepMind
zcheng@google.com

ABSTRACT

Learning expressive representations for high-dimensional yet sparse features has been a longstanding problem in information retrieval. Though recent deep learning methods can partially solve the problem, they often fail to handle the numerous sparse features, particularly those tail feature values with infrequent occurrences in the training data. Worse still, existing methods cannot explicitly leverage the correlations among different instances to help further improve the representation learning on sparse features since such relational prior knowledge is not provided. To address these challenges, in this paper, we tackle the problem of representation learning on feature-sparse data from a graph learning perspective. Specifically, we propose to model the sparse features of different instances using hypergraphs where each node represents a data instance and each hyperedge denotes a distinct feature value. By passing messages on the constructed hypergraphs based on our Hypergraph Transformer (HyperFormer), the learned feature representations capture not only the correlations among different instances but also the correlations among features. Our experiments demonstrate that the proposed approach can effectively improve feature representation learning on sparse features.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; *Machine learning*; • **Information systems** → *Information retrieval*.

KEYWORDS

Sparse Features; Hypergraph; Graph Neural Networks

This work was mainly done during Kaize's internship at Google.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGIR '23, July 23–27, 2023, Taipei, Taiwan
© 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/10.1145/3539618.3591999

ACM Reference Format:
Kaize Ding*, Albert Jiongqian Liang, Bryan Perrozi, Ting Chen, Ruoxi Wang, Lichan Hong, Ed H. Chi, Huan Liu, and Derek Zhiyuan Cheng. 2023. HyperFormer: Learning Expressive Sparse Feature Representations via Hypergraph Transformer. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3539618.3591999

1 INTRODUCTION

As one prevailing line of research for dealing with sparse features, researchers try to model the cross features in either the raw feature level [22] or the embedding level [31] to improve the representation expressiveness. Despite their success, existing methods still have a major bottleneck of capturing the following relational information within data: (1) *Instance Correlations*. Most of the existing efforts assume training instances are independently and identically distributed (i.i.d.), while different instances in the data may share correlated behavior/feature patterns [36]. Since the features of an instance could be extremely sparse, leveraging the knowledge from other instances is highly beneficial to improve the representation quality of sparse features in a collective way. Yet, how to model the correlations between different data instances without prior knowledge remains unexplored in this field; (2) *Feature Correlations*. Different feature values are usually correlated (e.g., two different feature values are commonly shared by many instances), which should not be neglected for learning expressive feature representations. In real-world systems, as the data scale grows continuously in a power-law distribution, a large number of features (a.k.a., tail features) appear few times in the training set. As a result, existing methods such as feature interaction learning approaches that rely on feature co-occurrence will lose their efficacy on tail features due to their rarity in the data [9]. How to better capture feature correlations is a key to improve the representation learning of sparse features. Based on those two limitations, one natural research question to ask is that – *given the input data with high-dimensional and sparse features, is there a natural way to explicitly model both instance correlations and feature correlations simultaneously*?

To answer this, we study relational representation learning on data with sparse features from a graph learning perspective. Due to the fact that the relationships among instances are naturally

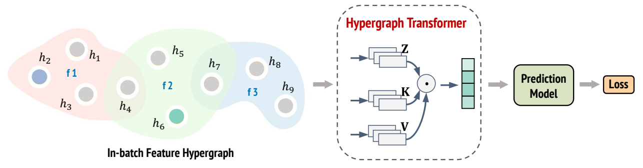

Figure 1: Illustration of the proposed HyperFormer.

high-order rather than pair-wise, e.g., a group of users sharing the same feature "location", instead of using simple graph, we adopt hypergraph [37] to model the high-order correlations among data instances. Specifically, we take feature values as the proxy (i.e., hyperedges) to connect different data instances (i.e., nodes) with the same feature values. To facilitate learning representations on each hypergraph constructed from the feature-sparse data, we develop a plug-and-play model – Hypergraph Transformer (HyperFormer), serving as an embedding module and is compatible to be trained together with various architectures and tasks. HyperFormer iteratively aggregates the information from hyperedges to nodes and vice versa, enabling multi-hop message-passing on the constructed hypergraphs. It captures the instance correlations as well as feature correlations simultaneously. The resulted feature representations can improve the predictive power of different models on feature-sparse data. Experiments on (i) CTR prediction and (ii) top-K item recommendation tasks demonstrate that HyperFormer is highly generalizable, and further enhances SOTA art approaches of representation learning for sparse features.

2 METHODOLOGY

Problem Definition. For the sake of simplicity, we only consider sparse features and use multi-hot representation for sparse features, where each sparse feature is also called a field. The input of our problem is a high-dimensional sparse vector $\mathbf{x} \in \mathrm{R}^N$ of multi-hot representation, where N is the total number of sparse feature values. Also, $x_i = 0$ means the i-th feature value does not exist in the instance and $x_i = 1$ means otherwise. The objective is to learn a low-dimensional embedding vector $\mathbf{e} \in \mathrm{R}^d$ that represents the raw input features in the latent space.

Existing works apply an embedding layer to project the input features into a low dimensional feature vector, which is commonly implemented by looking up from an embedding table $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_N] \in \mathbb{R}^{N\times d}$ and concatenating the retrieved embeddings into a dense real-value vector. Correspondingly, $\mathbf{f}_k$ can be regarded as the dense representation for feature x_k. In this paper, we argue that existing methods cannot explicitly consider the correlations between instances and the correlations between features, leading to the feature representations being less expressive.

Learning with Sparse Features has been a classic yet challenging problem in information retrieval and recommender system. A prevailing line of research tries to model the cross features in either the raw feature level or the embedding level. Compared to conventional approaches [13, 20, 22], deep learning models have shown their superiority for handling high-dimensional sparse features [2, 3, 23, 31]. Methods such as Wide&Deep [2], Deep Crossing [23], PNN [21], DCN [30], AutoInt [24], Fi-GNN [16], DCN-v2 [31] have been proposed to automatically model the cross features. However, existing methods are not able to explicitly capture the correlations between instances and are also ineffective to handle the tail features that appear rarely in the data.

Graph Neural Networks (GNNs) generally follow the neighborhood aggregation scheme [7, 10, 14, 26], learning the latent node representation via message passing among local or high-order neighbors in the graph [4, 6, 15]. More recently, GNNs have been actively explored to improve the performance of CTR prediction [9, 16, 17, 36] by capturing the interactions between features. Our approach leverages the idea of hypergraph [5, 8, 12, 27, 28, 37] and we build **HyperFormer** to perform representation learning on sparse features.

2.1 Feature Hypergraph

In this paper, we propose to alleviate the feature sparsity issue through relational representation learning. Since each specific feature can appear in multiple data instances, it can be naturally utilized as a bridge to capture instance correlations as well as feature correlations. For example, a group of users sharing the same feature values for "location" or "age". Such correlations among instances are inherently high-order rather than pair-wise, thus we propose to build *feature hypergraph* to model the input data and try to enable message-passing on it to capture desired relational information. Specifically, we define the feature hypergraph as follows:

Definition 2.1. **Feature Hypergraph**: A feature hypergraph is defined as a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \dots, v_n\}$ represents the set of nodes in the graph, and $\mathcal{E} = \{e_1, \dots, e_m\}$ represents the set of hyperedges. Specifically, each node represents a data instance and each hyperedge represents a unique feature value. Correspondingly, for any hyperedge e, it can connect arbitrary number of nodes/instances (i.e., $\sigma(e) \geq 1$).

Scalability Extension. Considering the fact that the scale of training data could be extremely large in practice, it is almost impossible to build a single feature hypergraph to handle all the data instances. To counter this issue, we propose to construct in-batch hypergraph based on data instances in the batch to further support mini-batch training. In Figure 1, we illustrate the steps for constructing the in-batch hypergraphs. For each batch, we randomly sample a batch of instances and update the hypergraph structure based on the data samples in the batch. From our experiments, the in-batch feature hypergraph is also effective for capturing the desired data dependencies and achieves satisfying performance improvements.

2.2 Hypergraph Transformer

To support representation learning on the constructed feature hypergraphs, we further propose a new model Hypergraph Transformer (HyperFormer) in this paper, which adopts the Transformer-like architecture [25] to exploit the hypergraph structure to encode both the instance correlations and feature correlations. Apart from conventional GNN models, each layer in HyperFormer learns the representations with two different hypergraph-guided message-passing functions, capturing high-order instance correlations and feature correlations simultaneously. Formally, a Transformer layer can be defined as:

$$\begin{aligned}
\mathbf{H}^l &= \mathrm{TF}_{edge}\Big(\mathbf{Q}^l_{edge} = \mathbf{H}^{l-1}, \mathbf{K}^l_{edge} = \mathbf{F}^{l-1}, \mathbf{V}^l_{edge} = \mathbf{F}^{l-1}\Big), \\
\mathbf{F}^l &= \mathrm{TF}_{node}\Big(\mathbf{Q}^l_{node} = \mathbf{F}^{l-1}, \mathbf{K}^l_{node} = \mathbf{H}^l, \mathbf{V}^l_{node} = \mathbf{H}^l\Big),
\end{aligned} \tag{1}$$

where $\mathrm{TF}\Big(\mathbf{Q},\mathbf{K},\mathbf{V}\Big) = \mathrm{FFN}\Big[\mathrm{softmax}(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d}})\mathbf{V}\Big]$ denotes the Transformer-like attention mechanism. In essence, TF_{edge} is a message-passing function that aggregates information from hyperedges to nodes and TF_{node} is another message-passing function that aggregates information from nodes to hyperedges. Specifically, we first look up from the feature embedding table $\mathbf{F}$ to initialize hyperedge representations and the initial node representation of each instance is computed by concatenating all its feature representations. Without loss of generality, we describe the two message-passing functions in a single HyperFormer layer l as follows:

Feature-to-Instance Message-Passing. With all the hyperedges representations $\{\mathbf{f}_j^{l-1} | \forall e_j \in \mathcal{E}_i\}$, we first apply an feature-to-instance (edge-to-node) message-passing to learn the next-layer representation $\mathbf{h}_i^l$ of node v_i. Specifically, we set the node representation from the last HyperFormer layer $l-1$ as the query. The representations of the connected hyperedges can be projected into keys and values. Formally, the similarity between the query and key can be calculated as:

$$\alpha_{ij} = \frac{\exp((\mathbf{h}_i^{l-1}\mathbf{W}_{edge}^Q)^{\mathrm{T}}\mathbf{k}_j)}{\sum_{e_p \in \mathcal{E}_i} \exp((\mathbf{h}_i^{l-1}\mathbf{W}_{edge}^Q)^{\mathrm{T}}\mathbf{k}_p)}, \quad \mathbf{k}_p = \mathbf{f}_p^{l-1}\mathbf{W}_{edge}^K, \tag{2}$$

in which $\mathbf{W}_{edge}^K$ is the projection matrix for the key of the feature-to-instance transformer. Then the next layer node representation can be computed as:

$$\mathbf{h}_i^l = \sigma\Bigg(\sum_{e_j \in \mathcal{E}_i} \alpha_{ij}\mathbf{f}_j^{l-1}\mathbf{W}_{edge}^V\Bigg), \tag{3}$$

where σ is the non-linearity such as ReLU and $\mathbf{W}_{edge}^V$ is a trainable projection matrix for the value.

Instance-to-Feature Message-Passing. With all the updated node representations, we again apply an instance-to-feature (node-to-edge) message-passing based on the Transformer layer to learn the next-layer representation of hyperedge e_j. Similarly, this process can be formally expressed as:

$$\mathbf{f}_j^l = \sigma\Bigg(\sum_{v_k \in \mathcal{V}_j} \beta_{jk}\mathbf{h}_k^l\mathbf{W}_{node}^V\Bigg), \tag{4}$$

where $\mathbf{f}_j^l$ is the output representation of hyperedge e_j and $\mathbf{W}_{node}^V$ is the projection matrix. β_{jk} denotes the attention score of hyperedge e_j on node v_k, which can be computed by:

$$\beta_{jk} = \frac{\exp((\mathbf{f}_j^{l-1}\mathbf{W}_{node}^Q)^{\mathrm{T}}\mathbf{k}_k)}{\sum_{v_p \in \mathcal{V}_j} \exp(\mathbf{f}_j^{l-1}\mathbf{W}_{node}^Q)^{\mathrm{T}}\mathbf{k}_p)}, \quad \mathbf{k}_p = \mathbf{h}_p^l\mathbf{W}_{node}^K, \tag{5}$$

where $\mathbf{W}_{node}^Q$ and $\mathbf{W}_{node}^K$ are the projection matrices for the query and key of the instance-to-feature message-passing. By stacking multiple HyperFormer layers, we are able to capture high-order instance correlations and feature correlations. The feature representations learned from the last HyperFormer $\mathbf{F}^L$ can be directly plugged into any model architecture as the feature embedding layer and improve the prediction performance on the downstream tasks.

3 EXPERIMENTS

To evaluate the effectiveness of the proposed approach, we conduct our experiments on two real-world tasks that often suffer from the feature sparsity issue: (i) click-through-rate (CTR) prediction and (ii) top-k item recommendation.

Table 1: Dataset Statistics.

	Dataset	#Sample	#Field	#Feature
CTR Prediction	MovieLen-1M	0.94M	7	3,529
	Criteo	45.84M	39	998,960
	Dataset	#User	#Item	#Features
Item Recommendation	Amazon-Movie	19,873	10,176	8,504
	Bookcrossing	48,999	193,765	5,100

3.1 Task1: CTR Prediction

Click-through-rate (CTR) prediction is a task that predicts how likely a user is going to click an advertisement. Typically, an instance sample is represented by high-dimensional and sparse features, such as user profile, ad attributes, and contextual features such as time, platform, and geographic location. We first try to evaluate the effectiveness of HyperFormer for CTR prediction.

Datasets. For the task of CTR prediction, we adopt two public real-world benchmark datasets in which the features are extremely sparse and the statistics for those datasets can be found in Table 1. To adopt the **MovieLens-1M** dataset for CTR prediction, we follow [24, 31] to transform the original user ratings into binary values. The dataset is divided into 8:1:1 for training, validation, and testing, respectively. The **Criteo** dataset is widely adopted for CTR prediction that includes 45 million users' ad clicks on display ads over a 7-day period. As in [24, 31], we use the data from the first 6 days for training and randomly split the data from the last day into validation and test sets.

Baselines. We include the following baselines methods for CTR prediction: Logistic Regression (**LR**) and Factorization Machine (**FM**) [22]. Different neural extensions of FM, including Neural Factorization Machine (**NFM**) [11], **xDeepFM** [18], and **HoFM** [1]. **AutoInt** [24] is designed to automatically learn the feature interaction with self-attention. **DCN-v2** [31] is an improved deep & cross network that models the explicit and bounded-degree feature interactions. To show the flexibility and effectiveness of HyperFormer, we integrate it into two representative baselines AutoInt and DCN-v2 then report their performance in the experiments.

General Comparison. We evaluate the performance of different methods based on two widely-used metrics for CTR prediction: AUC and LogLoss in Table 2. FM is able to model the second-order feature interaction and thus outperforms LR, which can only learn from raw feature input. With the power of deep neural networks, xDeepFM and NFM can improve the performance of FM in both datasets by incorporating non-linear transformations and interactions among features. AutoInt further improves on NFM by adaptively modeling feature interactions using an attention mechanism. DCN-v2 is also shown to be an effective approach for CTR. More importantly, our experimental results demonstrate the effectiveness of HyperFormer, as it improves the performance of the two representative CTR models for both AUC and LogLoss.

Further Analysis on Tail Features. Feature values usually follow a power-law distribution and those tail features only appear a few times among all the data examples. Without enough learning signals, it is hard for the low-frequency features to obtain informative

Table 2: AUC and Logloss on CTR prediction.

Method	Movielens-1M AUC	Movielens-1M LogLoss	Criteo AUC	Criteo LogLoss
LR	0.7716	0.4424	0.7820	0.4695
FM	0.8252	0.3998	0.7836	0.4700
NFM	0.8357	0.3883	0.7957	0.4562
xDeepFM	0.8286	0.4108	0.8009	0.4517
HoFM	0.8304	0.4013	0.8004	0.4508
AutoInt	0.8456	0.3797	0.8061	0.4455
DCN-v2	0.8402	0.3811	0.8045	0.4462
AutoInt+HyperFormer	**0.8462**	**0.3770**	**0.8072**	**0.4444**
DCN-v2+HyperFormer	**0.8471**	**0.3755**	**0.8061**	**0.4453**

embeddings, resulting in low CTR prediction accuracy for data samples that contain those low-frequency features. Our HyperFormer is proposed to address this issue by modeling the correlations between features through hypergraph message passing. To examine whether HyperFormer achieves this goal, we first sort and slice all the feature values in MovieLens-1M by frequency. Then we retrieve the test inputs that contain each set of feature values for evaluation. We compare the CTR performance of DCN-v2 with and without HyperFormer in Figure 2. Our results show that HyperFormer enables DCN-v2 to achieve better performance on samples with low-frequency features, as measured by both LogLoss and AUC. This demonstrates that HyperFormer is effective in enhancing the quality of feature embeddings for tail features, leading to more accurate CTR predictions for items with tail features.

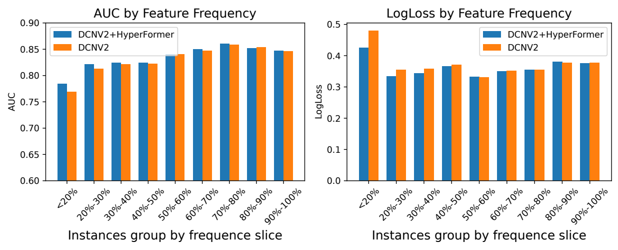

Figure 2: CTR performance across different instance groups.

3.2 Task2: Top-K Item Recommendation

In many real-world recommendation systems, the goal is to retrieve the Top-K relevant items for a given user. Indeed, the input of these top-k recommendation frameworks also suffers from the same issue as the CTR task in that the user and item features are also extremely sparse and follow the long-tailed distribution. To further demonstrate the generalizability of HyperFormer, we evaluate its performance on the relational representation learning for recommendation systems.

Datasets. In the experiment, we adopt two benchmark datasets for evaluation. **Amazon-Movie** consists of product reviews and metadata for the "Movie" category spanning from May 1996 to July 2014 [19]. **Bookcrossing** [38] collects the user-item ratings within the community, including both user demographic and age as well as item features such as Title, Author, Year, and Publisher. After removing the inactive users and items, we obtain the final datasets as summarized in Table 1. We randomly sample 70% data for model training, 10% for validation and 20% for testing.

Baselines. Due to its high efficiency and flexibility, **Two-tower** model with separate user and item towers is widely adopted as the fundamental learning architecture for large-scale top-k item retrieval [29, 32–35]. Specifically, the high-dimensional user and item features are input to the corresponding towers, and the preference scores are typically computed by a dot-product between user and item embeddings encoded by the corresponding towers. To solve the feature imbalance issue, **DAT** [34] was proposed to extend each tower with an extra learnable vector to memorize the cross-tower information. Recently, Yao et.al proposed **SSL** [32] to leverage latent feature correlations in a two-tower model by augmenting the data and incorporating an auxiliary self-supervised learning task.

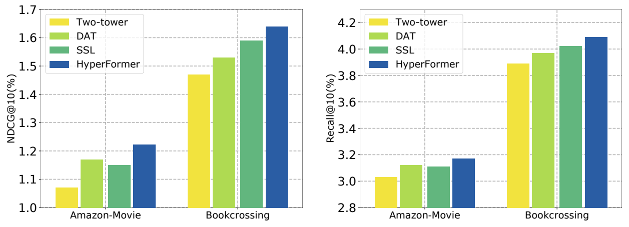

Figure 3: Top-k Recommendation performance comparison.

General Comparison. To evaluate HyperFormer on top-K item recommendation, we plug it into a two-tower recommendation model and compare it with the baseline methods above that were proposed to address the feature sparsity issue. We use NDCG@10 and Recall@10 as evaluation metrics and summarize the results for both Amazon-Movie and Bookcrossing in Figure 3. By introducing the category alignment loss and an extended vector capturing the cross-tower information, DAT can significantly improve two-tower in top-k recommendation for both datasets. We find that SSL significantly outperforms DAT in Bookcrossing but falls behind DAT in Amazon-Movie. However, the proposed HyperFormer consistently outperforms all other methods in both datasets, showcasing its effectiveness in feature representation learning. The advantage across both CTR prediction and top-k recommendation highlights the generalizability of HyperFormer and its potential to address feature sparsity in various real-world tasks.

4 CONCLUSION

In this paper, we focus on the problem of representation learning on high-dimensional sparse features. We propose to build feature hypergraphs to model the instance correlations and feature correlations explicitly. The proposed Hypergraph Transformer further enables message-passing on the constructed feature hypergraphs, resulting in more informative feature representations that encode instance correlations and feature correlations within the data. The evaluation of different methods on click-through-rate prediction and item recommendation demonstrate the effectiveness of our approach in capturing the relational information within data for learning informative feature representations.

REFERENCES

[1] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. Higher-order factorization machines. *NeurIPS*, 2016.

[2] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *RecSys*, 2016.

[3] Kaize Ding, Yichuan Li, Jundong Li, Chenghao Liu, and Huan Liu. Feature interaction-aware graph neural networks. *arXiv preprint arXiv:1908.07110*, 2019.

[4] Kaize Ding, Jianling Wang, James Caverlee, and Huan Liu. Meta propagation networks for few-shot semi-supervised learning on graphs. In *AAAI*, 2022.

[5] Kaize Ding, Jianling Wang, Jundong Li, Dingcheng Li, and Huan Liu. Be more with less: Hypergraph attention networks for inductive text classification. In *EMNLP*, 2020.

[6] Kaize Ding, Yancheng Wang, Yingzhen Yang, and Huan Liu. Eliciting structural and semantic global knowledge in unsupervised graph contrastive learning. In *AAAI*, 2023.

[7] Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. Data augmentation for deep graph learning: A survey. *SIGKDD Explorations*, 2022.

[8] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *AAAI*, 2019.

[9] Wei Guo, Rong Su, Renhao Tan, Huifeng Guo, Yingxue Zhang, Zhirong Liu, Ruiming Tang, and Xiuqiang He. Dual graph enhanced embedding neural network for ctr prediction. In *KDD*, 2021.

[10] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

[11] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *SIGIR*, 2017.

[12] Ujun Jeong, Kaize Ding, Lu Cheng, Ruocheng Guo, Kai Shu, and Huan Liu. Nothing stands alone: Relational fake news detection with hypergraph neural networks. In *BigData*, 2022.

[13] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. Field-aware factorization machines for ctr prediction. In *RecSys*, 2016.

[14] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[15] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*, 2019.

[16] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *CIKM*, 2019.

[17] Zekun Li, Shu Wu, Zeyu Cui, and Xiaoyu Zhang. Graphfm: Graph factorization machines for feature interaction modeling. *arXiv preprint arXiv:2105.11866*, 2021.

[18] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *KDD*, 2018.

[19] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, 2015.

[20] Junwei Pan, Jian Xu, Alfonso Lobos Ruiz, Wenliang Zhao, Shengjun Pan, Yu Sun, and Quan Lu. Field-weighted factorization machines for click-through rate prediction in display advertising. In *TheWebConf*, 2018.

[21] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. Product-based neural networks for user response prediction. In *ICDM*, 2016.

[22] Steffen Rendle. Factorization machines. In *ICDM*, 2010.

[23] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 255–262, 2016.

[24] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *CIKM*, 2019.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

[26] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[27] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. Next-item recommendation with sequential hypergraphs. In *SIGIR*, 2020.

[28] Jianling Wang, Kaize Ding, Ziwei Zhu, and James Caverlee. Session-based recommendation with hypergraph attention networks. In *SDM*, 2021.

[29] Jinpeng Wang, Jieming Zhu, and Xiuqiang He. Cross-batch negative sampling for training two-tower recommenders. In *SIGIR*, 2021.

[30] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *KDD*. 2017.

[31] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *TheWebConf*, 2021.

[32] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix Yu, Ting Chen, Aditya Menon, Lichan Hong, Ed H Chi, Steve Tjoa, Jieqi Kang, et al. Self-supervised learning for large-scale item recommendations. In *CIKM*, 2021.

[33] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *RecSys*, 2019.

[34] Yantao Yu, Weipeng Wang, Zhoutian Feng, and Daiyue Xue. A dual augmented two-tower model for online large-scale recommendation. *DLP-KDD Workshop*, 2021.

[35] Yin Zhang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Lichan Hong, and Ed H Chi. A model of two tales: Dual transfer learning framework for improved long-tail item recommendation. In *TheWebConf*, 2021.

[36] Zuowu Zheng, Changwang Zhang, Xiaofeng Gao, and Guihai Chen. Hien: hierarchical intention embedding network for click-through rate prediction. In *SIGIR*, 2022.

[37] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. *NeurIPS*, 2006.

[38] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *WWW*, 2005.