Click Through Rate Prediction on GNNs

Saw Nay Htet Win [1], Phaphontee Yamchote[1], Thanapon Noraset[1], Chainarong Amornbunchornvej[2],

[1] Mahidol University, Faculty of ICT, Nakhon Pathom,
nayhtetwin.saw@student.mahidol.ac.th
phaphontee.yam@student.mahidol.ac.th
thanapon.nor@mahidol.ac.th
[2] National Electronics and Computer Technology Center(NECTEC),
chainarong.amo@nectec.or.th

Abstract. Click-through rate (CTR) prediction Click-through rate (CTR) prediction is essential in online advertising and recommendation systems, affecting user engagement and revenue. Traditional methods like linear models and deep learning often struggle to effectively capture complex, high-order feature interactions. Graph Neural Networks (GNNs) offer improved performance by modeling these intricate relationships but face challenges regarding computational efficiency and interpretability, affecting scalability. We propose a new static graph-based GNN architecture for CTR prediction, incorporating advanced feature selection to enhance representation and reduce complexity. Our model employs a bilinear interaction mechanism to efficiently prioritize high-order interactions. Experiments on benchmark datasets show our approach achieves near state-of-the-art results, surpassing traditional deep learning models.

Keywords: Click-through Rate (CTR) Prediction, Graph Neural Networks (GNNs), Sparse feature selection, Online advertising, Recommendation systems

1 Introduction

Click-through rate (CTR) prediction is crucial for online advertising and recommendation systems, directly impacting user engagement and revenue through improved targeting and personalization [3, 16]. Predicting CTR accurately is challenging due to high-dimensional, sparse feature interactions, prompting the exploration of advanced modeling techniques [14, 26]. Traditional methods like logistic regression and factorization machines, focusing on linear and second-order interactions, have limitations in capturing complex interactions [7, 8]. Deep learning approaches, such as cross field networks and attention mechanisms, have advanced modeling high-order interactions [4, 6], with recent architectures employing two-stream designs that combine MLPs and explicit interaction mechanisms [15].

Recent studies have integrated graph neural networks (GNNs) into CTR prediction to explicitly capture complex feature interactions. GNNs effectively model non-Euclidean relationships, structuring data as graphs to represent dependencies explicitly, beneficial for capturing intricate relationships among categorical and numerical features [11, 24]. By transforming tabular data into feature graphs, GNNs propagate and aggregate information across connected features, enhancing representation learning beyond traditional methods. For instance, Fi-GNN [11] and GraphFM [12] use graph attention and feature memory modules, respectively, to model interactions explicitly. Compared to models relying on feature independence, feature graphs allow adaptive learning of relationships, improving generalization for CTR tasks [5, 25].

Despite their advantages, GNNs in CTR prediction face challenges balancing computational efficiency with high-dimensional feature complexity. Researchers have addressed these issues by simplifying graph structures or optimizing feature sets for better interpretability and efficiency [20, 22]. Dynamic models like Gated Deep Cross Network (GDCN) [19] introduce mechanisms for dynamic interaction prioritization, but static graph-based models also benefit from advancements like node embeddings, higher-order message passing, and adaptive aggregation, further enhancing static graph expressiveness.

1.1 Feature Interaction Modeling for CTR

Click-through rate (CTR) prediction is vital in online advertising and recommendation systems, largely relying on effective feature interaction modeling. Traditional methods like Logistic Regression (LR) capture simple first-order interactions, while Factorization Machines (FM) model second-order interactions through embedding vector products. However, FM cannot efficiently represent complex higher-order interactions.

Variants such as Field-aware Factorization Machines (FFM) [8] introduce field-specific embeddings, and Attention Factorization Machines (AFM) [21] apply attention mechanisms to emphasize significant feature pairs. Yet, these methods still focus only on second-order interactions. Deep learning approaches address these limitations by modeling higher-order interactions. Models like DeepFM [7] and NFM combine FM-based structures with deep neural networks (DNNs), implicitly capturing linear and non-linear interactions. Although effective, the implicit nature of these methods reduces interpretability.

Explicit interaction modeling is explored in models like xDeepFM [1], which uses the Compressed Interaction Network (CIN) to capture interactions through outer product operations. However, such methods face scalability issues. Recent improvements, like DCNV2 [10], tackle these challenges by employing low-rank techniques for efficient feature interaction approximation.

1.2 Graph Neural Networks for Feature Interaction Modeling

Graph Neural Networks (GNNs) model feature interactions by treating features as nodes and interactions as edges in a graph, allowing structured representation of complex relationships. This approach is particularly effective for CTR prediction, as it naturally represents interactions among features. Fi-GNN [11] was an early application of GNNs for CTR tasks, using a fully connected graph of categorical feature fields. It employs gated graph neural networks (GGNNs) to iteratively update node states based on neighboring information and utilizes attention mechanisms to weight feature interactions. GraphFM extends Fi-GNN by combining factorization machine (FM) interactions with GNN aggregation methods, explicitly modeling both second-order and higher-order interactions. GraphFM introduces a learned weighted adjacency matrix to select meaningful interactions, reducing irrelevant noise, and uses multi-head attention to capture diverse feature relationships, improving representational quality.

1.3 Opportunities and Challenges in Leveraging GNNs for CTR Prediction

Despite their advantages, GNN-based methods face several challenges. Computational efficiency remains an issue, especially for large-scale CTR datasets with high-dimensional, sparse features. The iterative aggregation process in GNNs can be computationally intensive, particularly with dense graphs representing numerous interactions. Another challenge is effective graph construction. Fully connected graphs, used by methods like Fi-GNN, may introduce noise by modeling irrelevant interactions. Although GraphFM addresses this by dynamically learning graph structures, further research is needed to improve scalability and robustness. Interpretability is also crucial for practical use. While methods such as Fi-GNN and GraphFM provide some interpretability through attention mechanisms and weighted edges, explaining specific feature interactions clearly remains challenging. Future research might combine the interpretability of factorization machines (FM) with the expressive capabilities of GNNs. In summary, GNNs offer significant potential for CTR prediction through structured modeling of feature interactions. Continued advancements in graph construction, attention mechanisms, and interaction selection will help overcome current limitations, enhancing their applicability in real-world scenarios.

2 Our Model

In this chapter, we introduce SparseGateGNN, a novel Graph Neural Network (GNN) designed for efficiently model feature interactions. Our model integrates key components, including gated graph learning, residual connections, adaptive normalization, and a dedicated Graph Layer for cross-feature interactions. This architecture aims to create sparse, interpretable graph structures while maintaining high predictive performance. Our model extends traditional GNN architectures by introducing a learnable graph structure that adapts dynamically to feature embeddings. The model is structured into three main components:

(1) Feature embedding
(2) Gated graph learning
(3) A prediction layer with cross-feature interaction

These components collectively capture and process complex relationships between features.

2.1 Feature Embedding Layer

The feature embedding layer transforms raw input features into dense, low-dimensional vectors. We follow FiGNN Li et al. [11] to represent features in a continuous vector space where semantic relationships can be learned. Given input X with n features and an embedding dimension d, the output is E $\in$ R$n\times d$, where Ei represents the embedding of the i-th feature.

2.2 Gated Graph Learning Module

The gated graph learning module constructs a dynamic, sparse graph structure based on pairwise feature interactions. In this structure, edges represent interactions between features, allowing information to propagate and influence node representations. This module employs a gating mechanism to prioritize significant feature connections and includes steps for sparsity enforcement and normalization.

Pairwise Interaction Scores
Pairwise interaction scores G are computed using a learnable interaction matrix K: eq1

$$G = \sigma\,(KK^T), \tag{1}$$

where σ is the sigmoid function.

Sparsity Enforcement
A threshold τ is applied to enforce sparsity :

$$G'_{ij} = \begin{cases} G_{ij}, \ if\ G_{ij} > \tau \\ 0, \ otherwise, \end{cases} \tag{2}$$

Normalization
The adjacency matrix G is normalized to ensure numerical stability [17]:

$$G_{norm} = \ D^{-1/2}G'D^{-1/2}, \tag{3}$$

where D is the degree matrix. The resulting sparse graph Gnorm is then used for message passing and feature interaction.

2.3 Graphic Layer with Cross-Feature Interaction

A central component of SparseGateGNN is the Pairwise interaction scores, which integrates simple message passing and cross-feature interactions into the graph propagation process. This layer combines these mechanisms to refine feature embeddings and capture higher-order feature relationships critical for CTR prediction. Recent works Liu et al. [13], Zhang et al. [24] have demonstrated the effectiveness of incorporating GNN-based interaction modeling for CTR prediction.

Linear Message Passing

The Graph Layer applies a normalized adjacency matrix Gnorm to Propagate messages between neighboring features. This propagation is modulated by learnable weights to enhance feature embeddings:

$$H' = G_{norm} \cdot \mathrm{H}_{out,} \tag{4}$$

where $\mathrm{H}_{out} = \mathrm{W}_{out}$. H represents the transformed feature embeddings, and W_{out} is a learnable weight tensor.

Cross-Feature Interaction

To complement message passing, the Graph Layer computes cross-feature interactions by calculating pairwise relationships between feature embeddings. These interactions are captured as:

$$H_{cross} = \sum_{j=1}^{n} \mathrm{G}_{norm} \cdot \mathrm{C}_{ijk}, \tag{5}$$

where: i, j are node indices representing different entities in the graph. k is the feature dimension index, indicating the specific feature dimension of the interaction. Hi and Hj are the feature representations (embeddings) of nodes i and j, respectively. Wcross is a learnable weight matrix that governs the transformation of the feature interactions. The cross-interaction terms Ci jk are further scaled by the normalized adjacency matrix Gnorm to align the interactions with the graph's structure:

$$H_{cross} = \sum_{j=1}^{n} \mathrm{G}_{norm} \cdot \mathrm{C}_{ijk}, \tag{6}$$

Combining Message Passing and Cross-Feautre Interaction.

The outputs of message passing H′and cross-feature interactions Hcross are aggregated to capture both local and global feature relationships:

$$H_{combined} = H' + \mathrm{H}_{cross}, \tag{7}$$

Final Transformation

The combined outputs are processed through a learnable transformation and a non-linear activation function:

$$H^{final} = LN(LeakyReLU(\mathrm{W}_{in} \cdot \mathrm{H}_{combined} + b)), \tag{8}$$

where W_{in} and b are trainable parameters, and LeakyReLU introduces non-linearity. Layer normalization(LN) is applied to stabilize training and enhance model generalization. By integrating simple message passing with cross-feature interactions, the Graph Layer effectively captures both direct feature relationships and higher-order interactions. This combination enhances the representational power of the model, making it well-suited for tasks like CTR prediction.

2.4 Residual Learning and GRU Updates

To enhance feature representation, we adopt the following two components from FiGNN [11]: **Residual Connections:** These preserve the original feature information by adding input embeddings to the graph output:

$$H = E + H^{final}, \tag{9}$$

GRU-Based Updates: When enabled, gated recurrent units (GRUs) refine feature representations over multiple graph layers by modeling sequential dependencies:

$$H^t = GRU(H^{t-1}, H^{final}), \tag{10}$$

2.5 Transformer Pooling

The Transformer pooling mechanism leverages the Transformer encoder to enhance feature representation by modeling long-range dependencies among features. This approach integrates self-attention and feed-forward layers to refine feature embeddings before applying a pooling operation. Inspired by prior works Dwivedi and Bresson [5], Yun et al. [23] in Graph Transformers, we adapt this mechanism for CTR tasks, building on the foundations of the original Transformer model Vaswani [18]. Given input embeddings H $\in \mathrm{R}^{n \times d}$, where n is the number of features and d is the embedding dimension, the Transformer encoder processes the embeddings to capture intricate feature dependencies:

$$H_{transformed} = TransformerEncoder(H^t), \tag{11}$$

where the encoder consists of L layers, each including multi-head self-attention and feed-forward sublayers. The self-attention mechanism ensures that critical interactions between features are identified and preserved.
Pooling and Prediction. After transformation, mean pooling aggregates the feature representations across all dimensions:

$$H_{pooled} = \frac{1}{n}\sum_{i=1}^{n} H_{transformed,i}, \tag{12}$$

This pooled representation is passed through a fully connected layer to produce the final prediction:

$$\hat{y} = \sigma(\mathrm{W}_{pred} \cdot \mathrm{H}_{pooled} + b) \tag{13}$$

where σ is the sigmoid activation function, Wpred is the weight matrix, and b is the bias term.

The prediction layer uses a Transformer-based pooling mechanism to aggregate feature representations. The Transformer encoder captures long-range dependencies among features, and mean pooling aggregates the information across all dimensions. The model is trained using binary cross-entropy (BCE) loss for CTR prediction, with an additional $L1$ penalty on the adjacency matrix to encourage sparsity.

3 Experiments

In this section, we describe the experimental setup and results for evaluating the proposed SparseGateGNN model. We perform experiments on the widely used Criteo dataset for click-through rate (CTR) prediction under multiple configurations to validate the effectiveness and robustness of our model.

3.1 Experiment Setup

We use the Criteo[9] dataset and Avazu[2] Dataset, standard benchmark for CTR prediction tasks. These datasets contain millions of user-item interactions, with dense and sparse features, making it suitable for evaluating feature interaction models. Those datasets are preprocessed following common practices, including feature normalization and transformation into a format compatible with SparseGateGNN. Base configuration for SparseGateGNN is provided in Table 1. This serves as the reference setup for comparing different experimental variations.

3.2 Main Results

In our experiments, we evaluate the performance of our proposed SparseGateGNN model for click-through rate (CTR) prediction. The key finding is that SparseGateGNN achieves an AUC score of 0.8132 while utilizing only approximately 33% of the connections in the graph. This demonstrates the model's ability to effectively reduce computational complexity while maintaining competitive performance. Can see the performance comparison in Table 2. Among graph-based models for CTR prediction, SparseGateGNN outperforms existing approaches, highlighting the effectiveness of gating mechanisms. However, when compared to other state-of-the-art models outside the graph domain, it does not achieve the highest AUC. Nevertheless, the AUC score remains close to top-performing models, indicating that SparseGateGNN provides a promising trade-off between model complexity and predictive accuracy.

Table 1. Base Configuration for SparseGateGNN

Hyperparameter	Value	Hyperparameter	Value
Dataset ID	criteo_x1	Use Residual Connections	True
Task	Binary Classification	Use GRU	False
Loss Function	Binary Cross-Entropy	Reuse Graph Layer	False
Metrics	Logloss, AUC	Embedding Regularizer	1×10^{-8}
Optimizer	Adam	Max Gradient Norm	1.0
Learning Rate	1×10^{-4}	Net Regularizer	0
Embedding Dimension	32	Epochs	100
Batch Size	512	Shuffle	True
GNN Layers	4	Seed	2019
Layer Normalization	True	Early Stopping Patience	4

Table 2. Performance Comparison Across Datasets

Class	Model	Criteo		Avazu	
		Logloss	AUC	Logloss	AUC
Graph Models	FiGNN[11]	0.4453	80.62	0.3825	77.62
	GraphFM[12]	0.4399	80.91	0.3781	77.98
	SparseGateGNN (ours)	0.4389	<u>81.32</u>	0.3749	<u>78.86</u>
Classical Models	LR[11]	0.4695	78.20	0.3964	75.60
	FM[11]	0.4700	78.36	0.3856	77.06
	AFM[11]	0.4584	79.38	0.3854	77.18
	FmFM[12]	0.4434	80.83	0.3859	77.46
Cross Field Network	CIN[11]	0.4517	80.09	0.3829	77.58
	CrossNet[12]	0.4591	79.07	0.3868	76.67
	DCNv3[10]	0.4358	**81.62**	0.3695	**79.70**
	GDCN[19]	0.4364	81.58	0.3739	79.05

SparseGateGNN outperforms other GNN-based models for CTR prediction in terms of AUC and logloss metrics. And also, performance is comparable to other sate of arts model. Table 2 summarizes the comparative performance.

Graph and Threshold.
The feature interaction graph, representing relationships with interaction scores, stabilizes by epoch 13, confirmed by decreasing differences from the final graph and application of a 0.65 threshold to highlight key interactions. SparseGateGNN dynamically selects relevant feature interactions, reducing graph connectivity by 33% while maintaining high accuracy and lowering computational overhead compared to models like Fi-GNN and GraphFM. This demonstrates the benefit of sparsity in CTR tasks.

While SparseGateGNN outperforms other GNN-based CTR models (Fi-GNN, GraphFM) in AUC and logloss, it doesn't match top deep learning models like DCNv3 and GDCN in AUC, though the difference is small, offering a good balance between complexity and performance. This trade-off is partly due to bilinear message passing and Transformer pooling, with ablation studies showing bilinear interaction as crucial for modeling high-order dependencies and Transformer pooling aiding in capturing long-range dependencies.

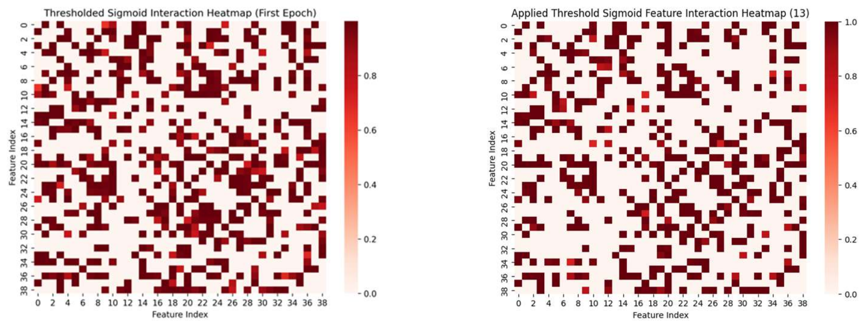

Fig. 1. Heatmap of k-interaction values learned by SparseGateGNN with 1st epoch left and 13th epoch right

3.3 Analysis

Ablation Study

To evaluate individual components of our model, we conducted an ablation study by selectively disabling features and comparing the resulting AUC scores (summarized in Table 3). Our full model, including L1 loss with self-loop, GRU, Transformer, and bilinear interaction, achieved the highest AUC of 0.8132. Removing Transformer pooling slightly reduced performance to 0.8124, underscoring its importance. Replacing GRU activation caused a decline to 0.8120, highlighting its role in sequential modeling. Omitting L1 loss and self-loop mechanisms further decreased the AUC to 0.8119, showing their contribution to node representation refinement. Excluding bilinear message passing resulted in the most substantial performance drop to 0.8084, emphasizing its critical role in capturing feature interactions. Overall, these results confirm the significant contribution of each component, particularly bilinear message passing and Transformer aggregation.

Table 3. Ablation study results showing the effect of disabling specific components on the model's AUC score of Criteo dataset

Model	AUC
Ours	**0.8132**
Ours – Transformer Pooling	0.8124
Ours - GRU Activation	0.8120
Ours - L1 and Self-loop	0.8119
Ours - Bilinear Message Passing	0.8084

Impact of Hyperparameter Settings

The impact of various hyperparameter settings was evaluated to understand their influence on model performance. Adding L1 regularization enhances graph sparsity, especially when used alongside self-loops, which alone also consistently boost performance. Thresholding methods showed minimal differences; however, fixed thresholding (e.g., 0.65) occasionally outperformed schedule-based approaches. Increasing the embedding regularizer value helped reduce overfitting and slightly improved test AUC. Lastly, larger embedding dimensions (e.g., 32) led to better AUC scores, albeit with a rise in model complexity.

Table 4. Experimental results for different hyperparameter settings.

Threshold	Regularizer	L1 loss	Self-loop	Dim	Parameters	AUC
Schedule	1.00×10^{-8}	No	No	16	33,553,665	0.8072
0.65	1.00×10^{-8}	No	No	16	33,553,665	0.8074
0.65	1.00×10^{-6}	Yes	No	32	67,440,001	0.8011
0.65	1.00×10^{-6}	No	Yes	32	67,440,001	0.8110
0.65	1.00×10^{-6}	Yes	Yes	32	67,440,001	0.8113
0.65	1.00×10^{-5}	Yes	Yes	32	67,440,001	0.8132

4 Conclusion

This study introduced SparseGateGNN, a novel GNN architecture for efficient and interpretable high-order feature interaction in CTR prediction. It combines gated graph learning, sparse selection, and transformer pooling, achieving an AUC of 0.8132 on benchmark datasets with only 33% of feature connections, demonstrating reduced overhead without sacrificing predictive power. Key components like bilinear message passing and transformer pooling proved significant. While outperforming other GNNs and competitive with state-of-the-art models, SparseGateGNN doesn't yet surpass top non-GNN models.

Limitations include the scarcity of diverse benchmark datasets, hindering generalization assessment, and the performance gap compared to leading CTR models. Future work includes exploring hybrid architectures (e.g., with DCN/DeepFM) and using meta-learning for dynamic graph optimization. Overall, graph-based approaches like SparseGateGNN show a promising balance of performance, efficiency, and interpretability.

Acknowledgments

This research project was partially supported by the **Faculty of Information and Communication Technology, Mahidol University**. The authors would like to express their sincere gratitude for the resources and support provided, which significantly contributed to the successful completion of this study.

References

1. [n. d.]. xDeepFM | Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. https://dl.acm.org/doi/abs/10.1145/3219819.3220023.
2. Avazu. 2015. Avazu Click-Through Rate Prediction. Kaggle. https://www.kaggle.com/c/avazu-ctr-prediction
3. Olivier Chapelle, Eren Manavoglu, and Romer Rosales. 2015. Simple and Scalable Response Prediction for Display Advertising. ACM Transactions on Intelligent Systems and Technology 5, 4 (Dec. 2015), 61:1–61:34. https://doi.org/10.1145/2532128
4. Wei Deng, Junwei Pan, Tian Zhou, Deguang Kong, Aaron Flores, and Guang Lin. 2021. DeepLight: Deep Lightweight Feature Interactions for Accelerating CTR Predictions in Ad Serving. In Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM '21). Association for Computing Machinery, New York, NY, USA, 922–930. https://doi.org/10.1145/3437963.3441727
5. Vijay Prakash Dwivedi and X. Bresson. 2020. A Generalization of Transformer Networks to Graphs. ArXiv (Dec. 2020).
6. Shereen Elsayed and Lars Schmidt-Thieme. 2023. Deep Multi-Representation
7. Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine Based Neural Network for CTR Prediction. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence. International Joint Conferences on Artificial Intelligence Organization, Melbourne, Australia, 1725–1731. https://doi.org/10.24963/ijcai.2017/239
8. Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-Aware Factorization Machines for CTR Prediction. In Proceedings of the 10th ACM Conference on Recommender Systems. ACM, Boston Massachusetts USA, 43–50. https://doi.org/10.1145/2959100.2959134
9. Criteo Labs. 2014. Criteo Display Advertising Challenge. Kaggle. https://www.kaggle.com/c/criteo-display-ad-challenge
10. Honghao Li, Yiwen Zhang, Yi Zhang, Hanwei Li, Lei Sang, and Jieming Zhu.2024. DCNv3: Towards Next Generation Deep Cross Network for CTR Prediction. https://doi.org/10.48550/arXiv.2407.13349 arXiv:2407.13349 [cs]
11. 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 Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM '19). Association for Computing Machinery, New York, NY, USA, 539–548. https://doi.org/10.1145/3357384.3357951
12. Zekun Li, Shu Wu, Zeyu Cui, and Xiaoyu Zhang. 2021. GraphFM: Graph Factorization Machines for Feature Interaction Modeling. ArXiv (May 2021).
13. Yuchen Liu, Chuanzhen Li, Han Xiao, and Juanjuan Cai. 2021. GCN-Int: A Click Through Rate Prediction Model Based on Graph Convolutional Network Interaction. IEEE Access 9 (2021), 140022–140030. https://doi.org/10.1109/ACCESS.2021.3116705
14. Fuyuan Lyu, Xing Tang, Dugang Liu, Liang Chen, Xiuqiang He, and Xue Liu. 2023. Optimizing Feature Set for Click-Through Rate Prediction. In Proceedings of the ACM Web Conference 2023 (WWW '23). Association for Computing Machinery, New York, NY, USA, 3386–3395. https://doi.org/10.1145/3543507.3583545
15. Kelong Mao, Jieming Zhu, Liangcai Su, Guohao Cai, Yuru Li, and Zhenhua Dong. 2023. FinalMLP: An Enhanced Two-Stream MLP Model for CTR Prediction. Proceedings of the

AAAI Conference on Artificial Intelligence 37, 4 (June 2023), 4552–4560. https://doi.org/10.1609/aaai.v37i4.25577

16. Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting Clicks: Estimating the Click-through Rate for New Ads. In Proceedings of the 16th International Conference on World Wide Web. ACM, Banff Alberta Canada, 521–530. https://doi.org/10.1145/1242572.1242643
17. Vijayalaxmi S Shigehalli and Vidya M Shettar. 2011. Spectral techniques using normalized adjacency matrices for graph matching. Int. J. Comput. Sci. Math 2, 4 (2011), 371–378.
18. A Vaswani. 2017. Attention is all you need. Advances in Neural Information Processing Systems (2017).
19. Fangye Wang, Hansu Gu, Dongsheng Li, Tun Lu, Peng Zhang, and Ning Gu. 2023. Towards Deeper, Lighter and Interpretable Cross Network for CTR Prediction. In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23). Association for Computing Machinery, New York, NY, USA, 2523–2533. https://doi.org/10.1145/3583780.3615089
20. Fangye Wang, Yingxu Wang, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, and Ning Gu. 2022. Enhancing CTR Prediction with Context-Aware Feature Representation Learning. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22). Association for Computing Machinery, New York, NY, USA, 343–352. https://doi.org/10.1145/3477495.3531970
21. Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua.2017. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence. International Joint Conferences on Artificial Intelligence Organization, Melbourne, Australia, 3119–3125. https://doi.org/10.24963/ijcai.2017/435
22. Ning Yin, Hongyan Li, and Hanchen Su. 2017. CLR: Coupled Logistic Regression Model for CTR Prediction. In Proceedings of the ACM Turing 50th Celebration Conference - China (ACM TURC '17). Association for Computing Machinery, New York, NY, USA, 1–9. https://doi.org/10.1145/3063955.3063976
23. Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim.2019. Graph Transformer Networks. In Advances in Neural Information Processing Systems, Vol. 32. Curran Associates, Inc.
24. Wei Zhang, Zhaobin Kang, Lingling Song, and Kaiyuan Qu. 2022. Graph Attention Interaction Aggregation Network for Click-Through Rate Prediction. Sensors 22, 24 (Jan. 2022), 9691. https://doi.org/10.3390/s22249691
25. Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph Neural Networks: A Review of Methods and Applications. AI Open 1 (Jan. 2020), 57–81. https://doi.org/10.1016/j.aiopen.2021.01.001
26. Kaixiong Zhou, Zirui Liu, Rui Chen, Li Li, Soo-Hyun Choi, and Xia Hu. 2022. Table2Graph: Transforming Tabular Data to Unified Weighted Graph. In Thirty-First International Joint Conference on Artificial Intelligence, Vol. 3. 2420–2426. https://doi.org/10.24963/ijcai.2022/336