Deep Graph Attention Neural Network for Click-Through Rate Prediction

Wen Fang[1] and Lu Lu[1,2*]
[1]School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
[2]Meizhou Modern Industrial Technology Research Institute, South China University of Technology, Meizhou, China
*Corresponding author email: lul@scut.edu.cn

***Abstract*—Click-through rate (CTR) prediction aims to estimate the probability of a user clicking on the item, which has critical importance both in advertising system and recommender system. Recently, deep learning-based methods have emerged due to its strong feature extraction ability. Learning user/item representations (aka. embeddings) is the core of these methods. However, these existing efforts pay little attention to encoding the relations among users and items in the embedding process, which limits the embedding effectiveness. In this paper, we propose a novel graph neural network framework for CTR prediction, namely the deep graph attention neural network (DGAN). Specifically, DGAN treats user-item interactions as a bipartite graph, which can naturally integrate node information and topological structure for modeling the relations. The key component of DGAN is attentive embedding propagation that recursively propagates embeddings from a node's neighbors to refine the node's embedding, and exploits graph attention mechanism to determine which neighbors to focus on. Comprehensive experiments are conducted on three public datasets and empirical results demonstrate DGAN achieves substantial gains compared with the mainstream models for CTR prediction.**

***Keywords*—Click-Through Rate Prediction, Recommender System, Deep Learning, Graph Neural Network, Embedding Propagation**

I. Introduction

Online advertising was born in the last century, which has become the major profit means of most internet companies. With the explosion of information on the internet, it is becoming increasingly important to explore the way to accurately and efficiently predict user behaviors with limited network resources. Click-through rate (CTR) refers to the ratio of ad clicks to ad impressions, which reflects user behaviors and serves as a key indicator to measure the advertising effectiveness. As such, predicting CTR becomes a critical task, which is beneficial to optimize marketing content, improve advertising effectiveness and ensure the quality of user experience.

Considering the superiority of deep learning, such as automatic high-order feature extraction, and inspired by its immense success in computer vision [1], speech recognition [2] and natural language processing [3], deep learning-based methods have been proposed to conduct CTR prediction task [4]–[6]. Compared with many previous works [7], [8], these deep learning-based methods can avoid a lot of manual

DOI reference number: 10.18293/SEKE2020-035

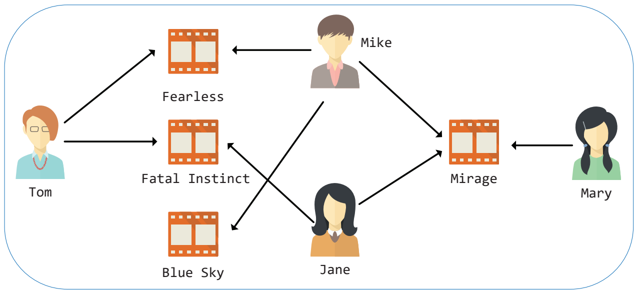


Fig. 1: An example of the user-item interaction graph.

feature engineering jobs and improve the model performance substantially.

Despite the great success achieved by these existing efforts, we argue that they are insufficient to learn effective embeddings for users and items. The key reason is that they treat each user-item interaction independently and overlook the latent relations in the interactions. For example, as shown in Figure 1, the path Mary $\rightarrow$ Mirage $\rightarrow$ Mike reveals the similar behavior between Mary and Mike, as they all have interacted with Mirage; the longer path Mary $\rightarrow$ Mirage $\rightarrow$ Mike $\rightarrow$ Blue Sky implies that Mary may be interested in Blue Sky, since her similar user Mike has ever watched Blue Sky. Furthermore, from the holistic view of a path length[1] of 3, Tom is more likely to have an interest in Mirage than Blue Sky, because there are two paths connecting Tom and Mirage, while only one path connects Tom and Blue Sky.

Being aware of aforementioned challenges and inspired by the wide success of leveraging graph neural networks [9]–[11], in this paper, we propose a novel graph neural network framework for CTR prediction, namely the deep graph attention neural network (DGAN). DGAN is a item-based model, which takes user behavior sequence and a candidate item as input, and outputs the probability of a user clicking the item. Specifically, we first represent user-item interactions as a bipartite graph, and then refine representation of each input item by recursively propagating the embeddings from its neighbors. Distinct from existing work [12], which treats

[1]In this paper, the length of a path refers to the number of edges it contains. For example, the length of path Mary $\rightarrow$ Mirage $\rightarrow$ Mike is 2 and the length of path Mary $\rightarrow$ Mirage $\rightarrow$ Mike $\rightarrow$ Blue Sky is 3.

neighbors equally, here we employ a graph attention mechanism to discriminate the importance of different neighbors. By doing so, latent relations among users and items are encoded in the process of embedding propagation.

After several embedding propagation layers, we obtain a refined representation vector for each item. To yield an adaptive representation of user interests with respect to current candidate item, we use an attention network to dynamically calculate the correlation between candidate item and historical behaviors, and take weighted sum to aggregate the user behavior sequence. User interests embedding and candidate item's embedding are finally fed into an interaction layer for CTR prediction.

The key contributions of this work are summarized as following:

- We emphasize the critical importance of encoding relations among users and items in the embedding process of deep learning-based models.
- We propose DGAN, a novel end-to-end CTR prediction framework based on graph attention network, which explicitly models relations among users and items by conducting embedding propagation and employs an attention module to capture the diversity characteristic of user interests.
- To validate the efficacy of DGAN, we conduct empirical studies on three real-world public datasets. The experimental results show our DGAN outperforms other mainstream models in the CTR prediction task.

The remaining parts of this paper are organized as follows. Section II reviews some related works. Section III describes proposed model DGAN in detail. Section IV presents the experiments. Section V summarizes this work and discusses future works.

II. Related Work

In this section, we mainly present recent studies of the CTR prediction and graph neural networks.

A. Click-Through Rate Prediction

Compared with traditional shallow learning, deep learning shows great potential on feature representation and combination. As such, more and more researchers apply deep learning to CTR prediction. Qu et al. [13] propose a product-based neural network (PNN) that combines factorization machine with multilayer perceptron (MLP). Cheng et al. [5] propose a novel structure Wide&Deep that cleverly fuses the linear model and the deep neural network. Guo et al. [14] propose a factorization-machine based neural network (DeepFM), which employs factorization machines and deep neural network to model low-order and high-order feature interactions. Shan et al. [15] propose a Deep Crossing model composed of an embedding layer, a stacking layer and a cascade of residual units for CTR prediction. Zhu et al. [16] propose a Deep Embedding Forest (DEF) based on Deep Crossing, which replaces the residual units in Deep Crossing by a forest layer and improves prediction efficiency by pre-training. Zhou et al. [17] propose a Deep Interest Network (DIN), which considers the lack of modeling of user behavior diversity and local activation in most CTR prediction studies. Zhou et al. [18] propose Deep Interest Evolution Network (DIEN) to model users' sequential behaviors, which enriches the representation of users and improves the prediction accuracy significantly. Feng et al. [19] propose a novel CTR model named Deep Session Interest Network (DSIN) that leverages users' multiple historical sessions in their behavior sequences. Despite the substantial gains achieved by these efforts, little attention has been paid to encoding relations among users and items in the embedding process, which degrades the model performance. In this paper, we employ graph neural network to fill this gap.

B. Graph Neural Networks

Graph Neural Network (GNN) is an extension of convolutional neural network to process underlying data with irregular structure (such as graphs). Kipf et al. [20] propose a convolutional architecture for semi-supervised learning on graph-structured. Velickovic et al. [21] propose an attention-based GNN architecture for node classification. Recently, researchers also apply GNN to recommender systems. Berg et al. [22] propose GC-MC, which employs GNN to learn representations of users/items on user-item graph, but only first-order neighbors are considered. Wang et al. [12] propose NGCF, which incorporates GNN into collaborative filtering and recursively performs propagation on user-item graph to capture the collaborative signal in high-order connectivity. However, neither GC-MC nor NGCF discriminate the importance of different neighbors. In this paper, we exploit the idea of graph attention network [21] to tackle this problem.

III. Methodology

In this section, we illustrate our proposed Deep Graph Attention Neural Network (DGAN) framework in detail, whose overall structure is shown in the Figure 2. We discuss the three main components: 1) embedding layer; 2) attentive embedding propagation layers; 3) attentive user interest extraction layer, respectively.

A. Embedding Layer

Embedding is a widely used technique to transform large scale sparse features into low-dimensional dense vectors, which has been used in many mainstream recommender models [12], [17]. With embedding, users can be represented as $E_u = [e_{u_1}, \cdots, e_{u_m}] \in R^{m \times d}$, where m is the number of users, d is the embedding size. Analogously, we can represent items as $E_i = [e_{i_1}, \cdots, e_{i_n}] \in R^{n \times d}$, where n is the number of items.

B. Attentive Embedding Propagation Layers

We take inspiration from the recent advance of GNNs [21], [23]. First, we introduce single layer propagation, and then exploit the idea of [12] to perform multiple layers propagation.

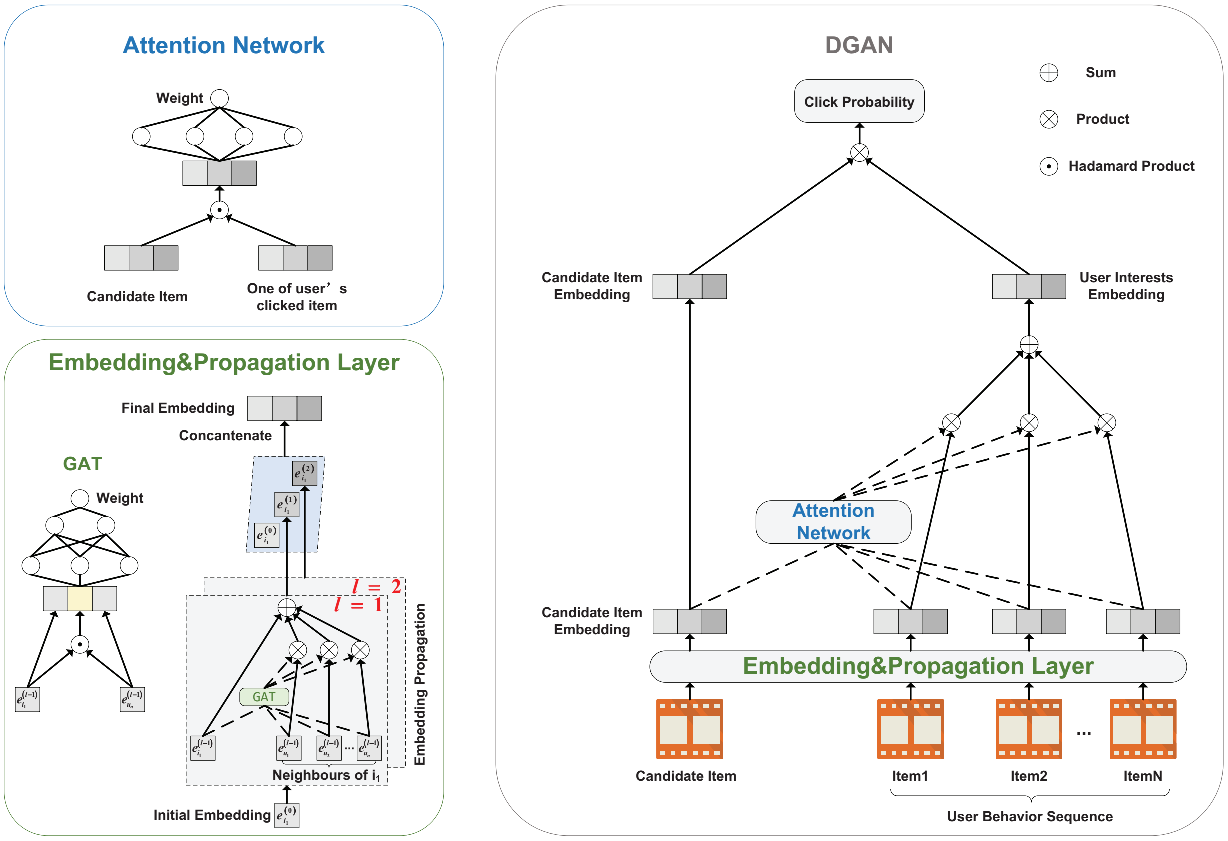


Fig. 2: An illustration of the DGAN model framework.

1) Single-layer Propagation: Obviously, histories of user-item interactions not only directly reflect a user's interests, but also represent the features of item to a certain extent [24]. We perform embedding propagation between user/item and its neighbors to encode these relations. For user-item interaction pair (u, i), embedding propagated from u to i is defined as:

$$m_{i \leftarrow u} = attn(e_i, e_u)e_u, \tag{1}$$

where $attn(\cdot)$ is the graph attention function that takes embeddings e_i and e_u as input and outputs the attention coefficients α_{iu}, reflecting the importance of local neighbors on e_i. Distinct from traditional graph attention network [21], which simply concatenates e_i, e_u and feeds it into a single-layer feedforward neural network to compute coefficients α_{iu}, we additionally model user-item interaction via $e_i \odot e_u$, where $\odot$ denotes the Hadamard product. Inspired by DKN [25] multi-channel mechanism, we treat user embedding e_u , item embedding e_i , user-item interaction embedding $e_i \odot e_u$ as three channels, and concatenate the three embedding matrices as

$$T_{iu} = [W_1 e_i || W_2(e_i \odot e_u) || W_3 e_u], \tag{2}$$

where $W_1, W_2, W_3 \in R^{d' \times d}$ are the trainable weight matrices. After getting the multi-channel input T_{iu}, to increase the learning ability, we feed it in double-layer feedforward neural network $\mathcal{D}$:

$$H_{iu} = \mathcal{D}(T_{iu}). \tag{3}$$

To make coefficients easily comparable across different nodes, we normalize them by adopting the softmax function:

$$\beta_{iu} = softmax(\alpha_{iu}) = \frac{exp(H_{iu})}{\sum_{k \in N_i} exp(H_{ik})}, \tag{4}$$

where N_i denotes the neighbors of item i. By assigning different neighbors with a different weight, the attention mechanism is capable of discriminating the importance of different neighbors, so as to precisely capture latent relations among users and items. Therefore, embedding propagated from user u to item i is implemented as:

$$m_{i \leftarrow u} = \beta_{iu} e_u. \tag{5}$$

Given the embeddings propagated from neighbors, we take weighted sum to aggregate them. Specifically, the aggregation function is expressed as:

$$e_i^{(1)} = LeakyReLU(m_{i \leftarrow i} + \sum_{u \in N_i} m_{i \leftarrow u}), \tag{6}$$

$$m_{i \leftarrow i} = W_1 e_i, \tag{7}$$

where $e_i^{(1)}$ denotes the refined embedding of item i after the first propagation layer. Note that we additionally aggregate the original embedding of item i via $W_1 e_i$ (W_1 is the trainable weight matrix defined in Equation 2), whose purpose is to retain the original features of item i. Analogously, we can obtain the refined embedding $e_u^{(1)}$ for user u.

2) Multi-layer Propagation: We argue that the first-order neighbors are not sufficient to encode relations among users and items. Therefore, based on the single-layer propagation, we follow a similar paradigm [12] to gather the embeddings propagated from the longer path neighbors. Specifically, the high-order propagation strategy is represented as follow:

$$e_i^{(l)} = LeakyReLU(m_{i \leftarrow i}^{(l)} + \sum_{u \in N_i} m_{i \leftarrow u}^{(l)}), \tag{8}$$

where l is path length. Similar to Equation 1 and 7, terms in Equation 8 can be expressed as:

$$m_{i \leftarrow u}^{(l)} = attn(e_i^{(l-1)}, e_u^{(l-1)}) e_u^{(l-1)}, \tag{9}$$

$$m_{i \leftarrow i}^{(l)} = W_1^{(l)} e_i^{(l-1)}, \tag{10}$$

where $W_1^{(l)} \in R^{d_l \times d_{l-1}}$ is the trainable transformation matrix, d_l is the dimension; $e_i^{(l-1)}$ is the representation of item i yielded from previous $(l-1)$ propagation layers, which further contributes to the representation of item i at layer l. Analogously, the representation for user u at the layer l can be obtained. Hereafter, high-order relation like path Mary $\rightarrow$ Mirage $\rightarrow$ Mike $\rightarrow$ Blue Sky can be extracted in the process of embedding propagation. Such high-order relation is crucial to encode the user's preference.

After propagating with l layers, we obtain a set of embeddings for item i, namely $\{e_i^{(0)}, e_i^{(1)}, \cdots, e_i^{(l)}\}$. Obviously, embeddings obtained from different propagation layers play a different role in representing item i. Towards this end, by exploiting the mechanism of layer-aggregation [23], we concatenate them to constitute the final embedding for item i:

$$e(i) = e_i^{(0)} || e_i^{(1)} || \cdots || e_i^{(l)}, \tag{11}$$

where $||$ is the concatenation operation. In this way, we encode relations among users and items in the embedding process, which not only enrich the initial embeddings, but also endow our model with powerful expressive capability.

C. Attentive User Interest Extraction

Given user u with behavior sequence $\{c_1^u, c_2^u, \cdots, c_{N_u}^u\}$, the embeddings of his clicked items can be expressed as $e(c_1^u), e(c_2^u), \cdots, e(c_{N_u}^u)$. To obtain a representation vector of user interests with respect to current candidate item, a general way [4], [5] is to process the list of embedding vectors via a pooling layer:

$$e(u) = pooling(e(c_1^u), e(c_2^u), \cdots, e(c_{N_u}^u)). \tag{12}$$

Average pooling seems a good choice to achieve above goal, since it simply conducts element-wise average operations of the list of embedding vectors. But the user interests embedding obtained this way remains the same for a specific user, no matter what candidate items are given. However, interests of user with rich behaviors are diverse, and user u's behaviors ought to have different effects on the candidate item t_j when predicting whether user u will click t_j. We model this process by using an attention network [17]. The attention network is illustrated in the left upper part of Figure 2. Specifically, for each item c_i^u clicked by user u and the candidate item t_j, we first conduct Hadamard product of their representation vectors, then feed it into a feed-forward network $\mathcal{G}$ and use softmax function to normalize the outputs:

$$w_i = softmax(\mathcal{G}(e(c_i^u) \odot e(t_j))). \tag{13}$$

The attention network takes the embedding of candidate item and a clicked item as input and outputs the impact weight. Then, we can obtain the final user interests embedding with respect to the candidate item by calculating the weighted sum of his clicked items embeddings:

$$e(u) = \sum_{i=1}^{N_u} w_i e(c_i^u). \tag{14}$$

Finally, given a user interests embedding $e(u)$ and candidate item embedding $e(t_j)$, we perform inner product to calculate the probability of a user u clicking the candidate item t_j:

$$p_{u,t_j} = e(u)^{\mathsf{T}} e(t_j). \tag{15}$$

D. Model Optimization

In view of the good performance of binary cross-entropy loss (aka. log loss) in deep recommender models, in this paper, we adopt it as the objective function to optimize model parameters, as follows:

$$Loss = \frac{-1}{N} \sum_{(u,t_j) \in S} (y_{u,t_j} \log p_{u,t_j} + (1 - y_{u,t_j}) \log(1 - p_{u,t_j})), \tag{16}$$

Where S is training set, N denotes the number of samples in S, p_{u,t_j} to be in (0, 1) represents prediction probability of a user u clicking the candidate item t_j, which is calculated by the current model parameters. Target value y_{u,t_j} is a binarized 1 or 0, which denotes whether u has interacted with t_j or not.

IV. Experiments

In this section, we present our experiments in detail. We start by introducing experimental datasets, baselines, evaluation metrics, parameter settings, then compare the proposed model with the baselines and analyze the results.

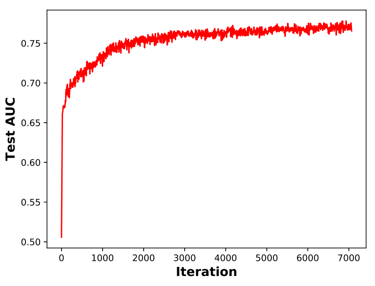
A. Dataset Description

Amazon(Electronics)[2]. Amazon Dataset is a widely used benchmark dataset in E-commerce, which consists of product reviews and metadata from Amazon. We use a subset called Electronics to conduct experiments.

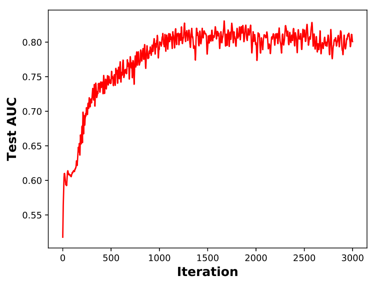
Amazon(Video Games)[3]. Video Games dataset is also a subset of Amazon, which contains rich user behaviors.

Yelp2018[4]. This dataset is adopted from the 2018 edition of the Yelp challenge. Here local businesses like restaurants and cinemas are treated as items. Specially, we take the subset where the timestamp is from Jun, 2017 to Jun, 2018.

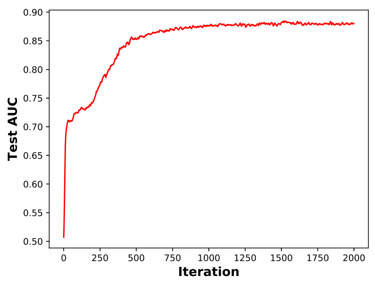
[2]http://jmcauley.ucsd.edu/data/amazon
[3]http://jmcauley.ucsd.edu/data/amazon
[4]https://www.yelp.com/dataset/challenge



(a) Electronics-AUC



(b) Video Games-AUC



(c) Yelp2018-AUC

Fig. 3: Testing performance of DGAN in each iteration.

To ensure the quality of datasets, we filter the original data which keeps each user with at least 10 interactions and generate negative samples which are of equal size with the positive ones. Samples in each dataset are split into 4:1 for training, test. The detailed characteristics of the three datasets are summarized in table I.

TABLE I: Statistics of datasets used in this paper.

Dataset	Users	Items	Samples
Amazon(Electronics)	45,225	61,918	1,547,004
Amazon(Video Games)	24,303	10,673	414,954
Yelp2018	25,937	92,249	796,994

B. Baselines

- **Wide&Deep [5]**. Wide&Deep is widely used in industrial applications, combining a (wide) linear part with a (deep) non-linear part.
- **PNN [13]**. PNN uses a product layer to capture interactive patterns between inter-field categories.
- **DeepFM [14]**. DeepFM is also a general deep model for recommendation, which employs factorization machines and deep neural network to model low-order and high-order feature interactions.
- **DIN [17]**. DIN fully considers user behavior diversity. By exploiting the idea of attention mechanism, it can learn the different representation of users' historical behaviors with respect to the candidate item.

C. Evaluation Metrics

In the experiment, we adopt AUC (Area Under ROC Curve) [26] to evaluate the performance of our framework and baselines. AUC is a widely used metric in CTR prediction field. It reflects the ranking ability of the model, defined as follows:

$$AUC = \frac{1}{m^+m^-} \sum_{x^+\in D^+} \sum_{x^-\in D^-} (I(p(x^+) > p(x^-))), \quad (17)$$

where D^+ is the set of all positive samples, D^- is the set of all negative samples, function $p(\cdot)$ outputs the predicted probability of the sample x, $I(\cdot)$ is the indicator function, m^+ is the size of D^+ and m^- is the size of D^-.

D. Parameter Settings

We implement our DGAN model in Tensorflow. The dimension of both user embeddings and item embeddings are set as 32. The number of propagation layers is set as 3. The learning rate is set as 0.01. The batch size is set as 512. We use Adam [27] to train DGAN by optimizing the log loss. The key parameter settings for baselines are as follows. The embedding size is fixed to 128 for all baselines, which results in better performance. Hyperparameters in the baselines are set the same as DGAN. Each experiment is repeated five times, and we report the average performance as results.

TABLE II: Results (AUC) on three public datasets.

Model	Electronics	Video Games	Yelp2018
Wide&Deep	0.7675	0.8137	0.8541
PNN	0.7681	0.8152	0.8597
DeepFM	0.7697	0.8173	0.8684
DIN	0.7712	0.8215	0.8692
DGAN	**0.7782**	**0.8302**	**0.8832**

E. Results

Table II reports the results of comparison of different models. Figure 3 shows the testing performance curve of DGAN. The major findings are summarized as below:

- Wide&Deep performs comparably poorly than other baselines. This indicates that manually designed features are insufficient to extract the representation of items.
- Compared with Wide&Deep, the performance of PNN verifies that automative high-order feature interactions can improve the representation learning ability.
- DeepFM generally performs better than PNN. Such improvement might be attributed to the combination of powerful factorization machines and a specially designed neural network.

- DIN generally achieves remarkable improvements since it uses attention mechanism to model user's diverse behaviors.
- DGAN consistently performs best in the three datasets. Specially, DGAN outperforms the strongest baselines with respect to AUC by 0.91%, 1.06%, and 1.61% in Electronics, Video Games, and Yelp2018, respectively. This is mainly because it considers the modeling of relations among users and items, which is overlooked in most click-through rate prediction studies.

V. Conclusion and Future Work

In this work, we propose a novel framework DGAN, which incorporates graph neural network into recommendation. DGAN addresses two major challenges on the CTR prediction task: 1) Distinct from traditional methods that encode user or item independently, DGAN fully takes relations among users and items into consideration by recursively propagating embeddings on user-item graph structure. 2) With respect to different candidate items, DGAN exploits an attention network to obtain an adaptive embedding vector of user interests. Extensive experiments are conducted on three datasets from Electronics, Video Games, and Yelp2018. The results demonstrate the rationality and efficacy of DGAN over several strong baselines.

In future, we plan to integrate knowledge graph and social networks into recommendation. This side information will be beneficial to understand user behaviors and improve recommendation interpretability. Moreover, with the great success of the Transformer for machine translation task in natural language processing, we will apply self-attention mechanism to investigate the sequential recommendation.

Acknowledgment

This research was supported by the National Nature Science Foundation of China (No. 61370103), Guangzhou Produce & Research Fund (201802020006) and Meizhou Produce & Research Fund (2019A0101019).

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[3] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, "A latent semantic model with convolutional-pooling structure for information retrieval," in *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, 2014, pp. 101–110.

[4] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 191–198.

[5] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st workshop on deep learning for recommender systems*, 2016, pp. 7–10.

[6] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & cross network for ad click predictions," in *Proceedings of the ADKDD'17*, 2017, pp. 1–7.

[7] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich, "Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine." Omnipress, 2010.

[8] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin *et al.*, "Ad click prediction: a view from the trenches," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 1222–1230.

[9] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*. Springer, 2018, pp. 593–607.

[10] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 974–983.

[11] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, and Z. Wang, "Knowledge-aware graph neural networks with label smoothness regularization for recommender systems," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 968–977.

[12] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 2019, pp. 165–174.

[13] Y. Qu, H. Cai, K. Ren, W. Zhang, Y. Yu, Y. Wen, and J. Wang, "Product-based neural networks for user response prediction," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 2016, pp. 1149–1154.

[14] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: a factorization-machine based neural network for ctr prediction," *arXiv preprint arXiv:1703.04247*, 2017.

[15] Y. Shan, T. R. Hoens, J. Jiao, H. Wang, D. Yu, and J. 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*, 2016, pp. 255–262.

[16] J. Zhu, Y. Shan, J. Mao, D. Yu, H. Rahmanian, and Y. Zhang, "Deep embedding forest: Forest-based serving with deep embedding features," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1703–1711.

[17] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1059–1068.

[18] G. Zhou, N. Mou, Y. Fan, Q. Pi, W. Bian, C. Zhou, X. Zhu, and K. Gai, "Deep interest evolution network for click-through rate prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5941–5948.

[19] Y. Feng, F. Lv, W. Shen, M. Wang, F. Sun, Y. Zhu, and K. Yang, "Deep session interest network for click-through rate prediction," *arXiv preprint arXiv:1905.06482*, 2019.

[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations*, 2017.

[21] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *6th International Conference on Learning Representations*, 2018.

[22] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.

[23] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," *arXiv preprint arXiv:1806.03536*, 2018.

[24] F. Xue, X. He, X. Wang, J. Xu, K. Liu, and R. Hong, "Deep item-based collaborative filtering for top-n recommendation," *ACM Transactions on Information Systems (TOIS)*, vol. 37, no. 3, pp. 1–25, 2019.

[25] H. Wang, F. Zhang, X. Xie, and M. Guo, "Dkn: Deep knowledge-aware network for news recommendation," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 1835–1844.

[26] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.