BaGFN: Broad Attentive Graph Fusion Network for High-Order Feature Interactions

Zhifeng Xie, Wenling Zhang, Bin Sheng, *Member, IEEE*, Ping Li, *Member, IEEE*, and C. L. Philip Chen, *Fellow, IEEE*

***Abstract*—Modeling feature interactions is of crucial significance to high-quality feature engineering on multi-filed sparse data. At present, a series of state-of-the-art methods extract cross features in a rather implicit bit-wise fashion and lack enough comprehensive and flexible competence of learning sophisticated interactions among different feature fields. In this paper, we propose a new Broad Attentive Graph Fusion Network (BaGFN) to better model high-order feature interactions in a flexible and explicit manner. On the one hand, we design an attentive graph fusion module to strengthen high-order feature representation under graph structure. The graph-based module develops a new bilinear-cross aggregation function to aggregate the graph node information, employs the self-attention mechanism to learn the impact of neighborhood nodes, and updates the high-order representation of features by multi-hop fusion steps. On the other hand, we further construct broad attentive cross module to refine high-order feature interactions at a bit-wise level. The optimized module designs a new broad attention mechanism to dynamically learn the importance weights of cross features and efficiently conduct the sophisticated high-order feature interactions at the granularity of feature dimensions. The final experimental results demonstrate the effectiveness of our proposed model.**

***Index Terms*—Feature interactions, graph neural networks, attention mechanism, broad learning system.**

I. Introduction

FEATURE engineering is a procedure of exploiting valid features from original data, which can significantly boost the performance of the predictive model. It is normally regarded as a central task to a variety of machine learning applications, such as recommendation system [1], computational advertising [2], search ranking [3] and so on. However, multi-field sparse data often fail to achieve ideal feature engineering because it is quite difficult to model intricate feature interactions effectively. To that end, previous feature interaction methods [4], [5] create cross features by manually exploiting multiple features. Such feature crossing is able to disclose the implicit relationships among multi-field sparse data. Nevertheless, the handcrafted feature interactions usually require extensive domain knowledge and consume a lot of time, which is hard to generate the appropriate cross features to a new domain. Therefore, in the process of feature engineering, it is crucial to automatically achieve high-order feature interactions from multi-field sparse data, which is a great help to improve the accuracy of predictive model.

Recently, a series of state-of-the-art methods have been proposed to generate cross features automatically from raw data. Factorization Machine (FM) [6] is regarded as a benchmark solution for multi-field sparse data in academia and industry due to its superior performance. Specifically, it transforms each feature into an embedding vector and learns second-order interactions via inner product of pairwise features. As an improved version of FM, Field-aware Factorization Machine (FFM) [7] takes into account feature interactions among different fields, which can yield more powerful performance. But the neglect of nonlinear and high-order feature interactions limits the prediction accuracy of these FM-based models. Later, numerous methods based on deep learning are proposed to probe the modeling of high-order and non-linear feature interactions, such as NFM (Neural FM) [8], Deep&Cross [2], DeepFM [9], xDeepFM [10], AutoInt (Automatic Feature Interaction) [11], FiBiNET (Feature Importance and Bilinear Feature Interaction Network) [12]. For example, NFM [8] and DeepFM [9] construct Deep Neural Networks (DNN) to model high-order interactions and meanwhile utilize FM to model low-order interactions. However, they extract cross features in a rather implicit bit-wise fashion, which often lacks good model explanations and will bring negative and uncontrollable effects due to some useless feature interactions.

Unlike these implicit methods, some optimized models are capable of learning high-order interactions explicitly or the importance of features by proposing their dedicated networks. For example, Deep&Cross [2] proposes a DNN-based cross network to learn certain bounded-degree feature interactions directly; xDeepFM [10] designs Compressed Interaction Network (CIN) to learn high-order feature interactions explicitly; AutoInt [11] constructs a self-attentive neural network to automatically learn different orders of feature interactions; FiBiNET [12] introduces Squeeze-Excitation Network (SENET) to learn the significance of features. Unfortunately, these op-

Manuscript received February 10, 2021; revised August 11, 2021; accepted September 26, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61872241 and Grant 61572316, in part by the National Key Research and Development Program of China under Grant 2019YFB1703600, and in part by The Hong Kong Polytechnic University under Grant P0030419, Grant P0030929, and Grant P0035358.

Z. Xie is with the Department of Film and Television Engineering, Shanghai University, Shanghai 200072, China; and also with the Shanghai Engineering Research Center of Motion Picture Special Effects, Shanghai 200072, China.

W. Zhang is with the Department of Film and Television Engineering, Shanghai University, Shanghai 200072, China.

B. Sheng is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (Email: shengbin@sjtu.edu.cn).

P. Li is with the Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong (Email: p.li@polyu.edu.hk).

C. L. P. Chen is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China; with the Navigation College, Dalian Maritime University, Dalian 116026, China; and also with the Faculty of Science and Technology, University of Macau, Macau 999078, China (Email: philip.chen@ieee.org).

timized methods are still not sufficiently effective and explicit since they just unite feature fields to learn interactions or introduce a single module to explore the importance of features. The amorphous combination or independent attention unit lacks the competence of learning sophisticated interactions among different feature fields in a flexible and explicit manner. To better model the sophisticated high-order interactions, Fi-GNN (Feature Interaction GNN) [13] first introduces Graph Neural Network (GNN) [14] to transform the modeling of feature interactions into the modeling of node interactions on a graph structure. Specifically, Fi-GNN is designed to model feature interactions by aggregating the graph node with its neighbors, and the edge weights reflect the importance of different node interactions. However, the simple aggregation manner limits the capability of cross features, and it is also incapable of learning the impact of aggregation of neighboring nodes on the node itself. In brief, the informative aggregation of graph nodes is not still comprehensive enough and flexible for sophisticated high-order interactions. Thus, we focus on how to take advantage of the GNN-based method and integrate some other advanced models to make up for its deficiency.

In this paper, a novel model named Broad Attentive Graph Fusion Network (BaGFN) is proposed to capture fine-grained cross features in a flexible and explicit manner. Concretely, inspired by Graph Convolutional Network (GCN) [15] and Broad Learning System (BLS) [16], the new model transforms features interactions into node interactions, and its architecture mainly consists of two key modules: attentive graph fusion module and broad attentive cross module. The first module constructs graph structure to generate feature representation by aggregating the information of first-order and second-order neighborhood nodes, and then employs the self-attention mechanism and multi-hop fusion steps to learn and update the high-order interactions of graph nodes. The module can not specify the significance of each interaction at the granularity of feature dimensions. To further refine the high-order interactions of the first module, the second module achieves a new broad attention mechanism, which can dynamically learn the weights of cross nodes which reflect the significance of different feature interactions on the final model prediction and quickly conduct high-order feature interactions at the granularity of feature dimensions. On the basis of the above two modules, our BaGFN model is able to capture the explicit influence of sophisticated high-order feature interactions among multiple fields, which will be beneficial to further improve the accuracy of the predictive model. To validate the effectiveness of our BaGFN model, we conduct extensive experiments on two benchmarks Click-Through Rate (CTR) prediction datasets (Avazu[1] and Criteo[2]) and a real-world industry dataset (Tobacco) with more feature fields. In summary, our main contributions are listed as follows:

- **A Broad Attentive Graph Fusion Network (BaGFN) for modeling high-order feature interactions.** The BaGFN model transforms features interactions into node interactions and extracts fine-grained cross features among multi-field sparse data in a flexible and explicit manner.
- **An attentive graph fusion module to strengthen high-order feature representation under graph structure.** The graph-based module designs a bilinear-cross aggregation function to aggregate the graph node information, employs self-attention mechanism to learn the impact of neighborhood nodes, and updates the high-order representation of features by multi-hop fusion steps.
- **A broad attentive cross module to refine high-order feature interactions at the bit-wise level.** The optimized module further designs a new broad attention mechanism to dynamically learn the importance weights of cross features and efficiently conduct the sophisticated high-order feature interactions at the bit-wise level.

[1]https://www.kaggle.com/c/avazu-ctr-prediction

[2]https://www.kaggle.com/c/criteo-display-ad-challenge

II. Related Work

Modeling Feature Interactions. Modeling feature interactions is an essential problem of multi-field sparse data and is studied extensively in the literature. A classic algorithm is FM [6], which pays attention to second-order feature interactions by capturing the inner-product of pairwise feature representations. Succeeding the success of FM, numerous variants of FM have been proposed to exploit more field information. FFM [7] presumes each feature field possesses a separate embedding vector for each of the other feature fields. Nevertheless, these FM-based methods only consider the second-order interactions of feature fields, which neglects the nonlinear and high-order feature interactions and limits the performance of their final predictive models.

In recent years, more and more methods based on deep learning has been proposed to model high-order feature interactions. Product-based Neural Network (PNN) [17] applies a product layer between embedding layer and DNN layer to learn second-order and high-order interactions. Different from it, Neural Factorization Machine (NFM) [8] proposes a bi-interaction pooling operation between the embedded layer and the neural network layer to model high-order and non-linear feature interactions. Meanwhile, some researchers focus on learning low-order and high-order feature interactions through a hybrid network. For instance, Wide&Deep [4] and DeepFM [9] have a shared feature embedding layer to its "wide" and "deep" component. Its "wide" part learns low-order interaction while the "deep" part models high-order interaction. However, these DNN-based methods use rather implicit bit-wise manners to conduct the high-order feature interactions, which may cause negative and uncontrollable cross features due to some useless feature interactions.

Besides, some other DNN-based methods utilize the specifically devised networks to capture high-order interactions in an explicit manner. Deep&Cross [2] introduces a novel cross network to capture the feature interactions of bounded degrees and the nonlinear feature interactions, which is further improved by [18] for more practical in large-scale industrial settings. xDeepFM [10] considers both the implicit and explicit high-order feature interactions, and devises CIN to model the feature interactions in an explicit manner and at a vector-wise level. Adaptive Factorization Network (AFN) [19] models

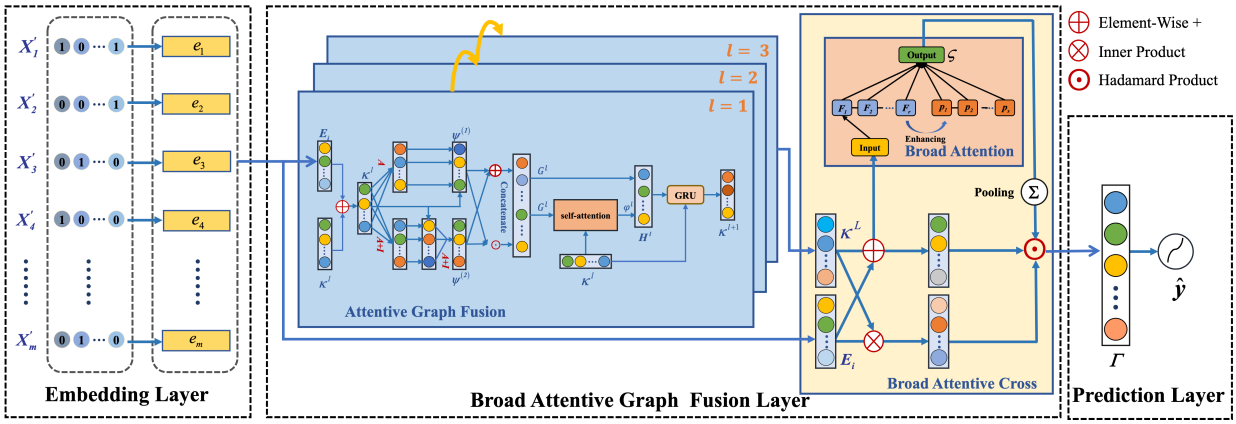


Fig. 1: The overall architecture of our BaGFN model. The new model consists of three layers: Embedding Layer, Broad Attentive Graph Fusion Layer, and Prediction Layer. As the core of the BaGFN model, the second layer consists of two key parts: attentive graph fusion module and broad attentive cross module. The first module strengthens high-order feature representation by integrating graph structure and self-attention mechanism. The second module designs a broad attention mechanism to refine high-order feature interactions at the bit-wise level.

the arbitrary-order cross features and their weights adaptively via a logarithmic transformation network. On the other hand, the attention mechanism is also introduced for multi-filed sparse data to improve the feature interactions by highlighting useful features. Attentional Factorization Machines (AFM) [20] introduces the method of attention mechanism to learn the importance of each second-order feature interaction based on FM. High-order Attentive Factorization Machine (HoAFM) [21] aggregates the representations of other co-occurred features and learns the different importance of co-occurred features on the granularity of dimensions. AutoInt [11] employs a multi-head self-attentive neural network with residual connections to model the different orders of feature interactions. FiBiNET [12] dynamically captures the significance of features via the Squeeze-Excitation Network. Dual Input-aware Factorization Machines (DIFM) [22] adaptively reweight the original feature representations at the bit-wise and vector-wise levels simultaneously. In these optimized methods, the amorphous combination or independent attention unit still lacks the competence of learning sophisticated interactions among different feature fields. Therefore, this paper will focus on how to model the sophisticated high-order interactions effectively in a more flexible and explicit manner.

Graph Neural Networks. Graph structure consists of nodes and edges, where the edge represents the relationship of connected nodes. In the past decade, a growing body of research is interesting in graphs representation, acquisition and applications [23]. Concretely, Xiao et al. [24] characterize graphs to measure similarity and clustering by computing permutation invariants from the heat kernel trace. Han et al. [25] construct a generative prototype for graphs by adopting a minimum description length approach. Bai et al. [26] propose two novel local-global nested graph kernels, which can simultaneously reflect the local and global graph characteristics in terms of the nested complexity traces. Recently, a number of deep learning algorithms have been applied on irregular inputs like graph structure data. Owing to the massively parallel architectures, modern GPUs (Graphics Processing Units) have been successfully utilized to accelerate the performance of graph computation [27]. Frasconi et al. [28] utilize recursive neural networks to represent and process data in graph. DeepWalk [29] uses truncated random walks to learn node latent representations in graph. Following the DeepWalk, LINE (Large-scale Information Network Embedding) algorithm [30] preserves both the local and global structural information to suit arbitrary types of information embedding network. Later, Cavallari et al. [31] consider embedding communities instead of individual nodes, which can identify the interaction between community embedding and detection as a closed loop, through node embedding. However, most of these shallow models cannot learn non-linear network structure, resulting in sub-optimal representations for large graphs.

GNN [32] is designed to address the above deficiencies, which is a type of neural networks that performs neural network operations via graph structure to acquire node representations. A typical GNN model consists of an iterative random-walk process in which node states are propagated by aggregating information from neighborhoods. Recently, numerous GNN variants with various types of novel aggregators and updaters have been proposed. For instance, GCN [15] tackles spectral information by employing the convolutional aggregator to operate the first-order neighborhood around each node. GraphSAGE (Graph Sample and Aggregate) [33] proposes three types of aggregators: mean aggregator, LSTM (Long Short-Term Memory) aggregator, and pooling aggregator. Graph Attention Networks (GAT) [34] captures different weights to different nodes in a neighborhood by stacking layers. Moreover, Cross-GCN [35] introduces a cross-feature operator to explicitly model the arbitrary-order cross features with complexity linear to feature dimension and order size.

Recently, GNN has been broadly applied to various tasks such as recommender system [1], action recognition [36], semantic segmentation [37], image recognition [38], visual question answering [39], and so on. Fi-GNN [13] first intro-

duces GNN to model feature interactions on a graph structure. But its simple aggregation manner limits the capability of cross features and it also can not learn the impact of aggregation among nodes. Thus our work attempts to deeply integrate GNN and attention mechanisms to model feature interactions of graph structure features for multi-field sparse data.

Broad Learning System. Due to the deep structure and abundant connecting weights, most networks based on deep learning usually put up with a time-consuming training process. In order to resolve this shortcoming of the long training process problem in the deep network, Pao and Takefuji [40] design a Random Vector Function-link Neural Network (RVFLNN). It transforms hidden nodes in the traditional Single Layer Feed Forward Neural Network (SLFN) into enhancement nodes and connects the output node with both input and enhancement nodes. Besides, the parameters transforming input nodes to enhancement nodes are randomly generated. This single change has greatly improved the performance of this network.

For the sake of resolving the high dimensional data problem and long training process, BLS [16] is proposed as a substitute for deep structure, which is inspired by RVFLNN and the dynamic step-wise updating algorithm [41]. BLS is designed as a flat network, where the original feature nodes are transferred as "mapped features", and the structure is expanded in a wide sense in the "enhancement nodes". Moreover, the connection weights are updated using ridge regression of the pseudoinverse method. Compared with deep learning methods, BLS can be easily constructed at a fast speed, even without a high-performance computer. In addition, due to the independence and flexibility of the "mapped features" and "enhancement nodes", there are many advanced variants of BLS [42] to improve the performance. In this paper, we introduce the BLS theory to construct a new broad attention module, which can achieve the efficient refinement of high-order feature interactions at a bit-wise level.

TABLE I: Notations of our BaGFN model.

Symbols	Definitions and Descriptions
m, M	the size of fields, the size of training instances
$X_m^{'}$	the one-hot encoding representation in the m-th field
X_i	the i-th instance with m field representations
E_i	the embedding output of X_i
n_p, n_q	the p-th and q-th graph nodes of E_i
$\xi(n_p, n_q)$	the edge weight between n_p and n_q
A	the adjacency matrix with $m \times m$ edge weights
l	the multi-hop feature fusion step
κ^l	the input features at the step l
$\mathcal{N}_p^{(1)}$, $\mathcal{N}_p^{(2)}$	the first-order and second-order neighborhood nodes of n_p
$\psi_p^{(1)}$, $\psi_p^{(2)}$	the first-order and second-order aggregations of n_p
G_p^l	the bilinear-cross aggregation of n_p at the step l
φ_p^l	the attention weight vector of n_p at the step l
H_p^l	the interaction representation of n_p at the step l
κ^L	the high-order feature representation after L steps
F^r, P^s	the mapping and enhanced feature nodes
ζ	the broad attentive weights of global-aware nodes
Γ	the final representation of cross features
$\hat{y}$, y	the predictive result, the ground truth
$\mathcal{L}$	the objective function with Log-likelihood Loss
W	all trainable parameters in our BaGFN model

III. Broad Attentive Graph Fusion Network

In this section, we will introduce the overview of our BaGFN model, which is illustrated in Fig. 1. The new architecture is composed of three components: (1) *Embedding Layer*, which converts the raw features of multi-field sparse data into the embedding vectors. (2) *Broad Attentive Graph Fusion Layer*, which is the core of our model. It contains two parts: attentive graph fusion module and broad attentive cross module. The first module integrates graph structure and self-attention mechanism to refresh high-order features representation. The second module further refines high-order feature interactions by designing a broad attention mechanism at the bit-wise level. (3) *Prediction Layer*, which combines the above cross features and then computes the final prediction probability. The mathematical notations used in this paper are summarized in Table I.

A. Embedding Layer

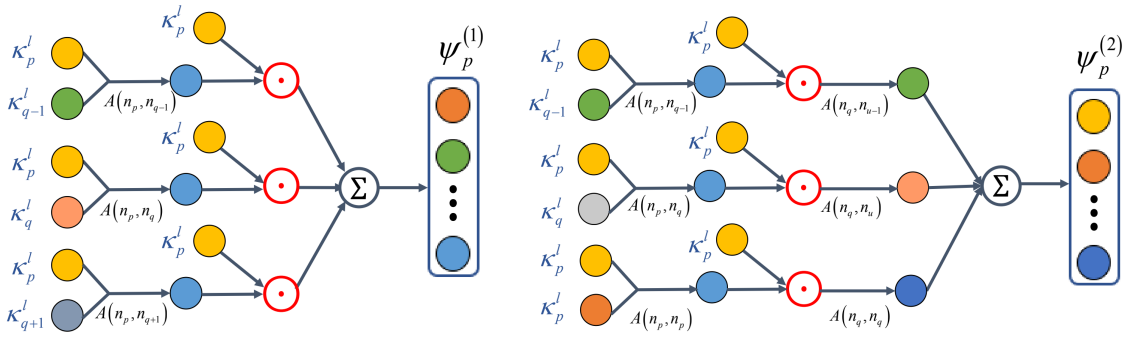
In the multi-field sparse data, each input instance is made up of several field-aware values, which include plenty of dynamic numerical fields and static categorical fields. For example, a movie instance has some categorical fields: {Language: Italian, Region: Italy, Director: Roberto Benigni, $\cdots$}, which can not be directly entered into a DNN-based model. Therefore, we should convert the instance into multi-field representations, including numerical representations from numerical fields and one-hot representations from categorical fields:

$$X_i = [X_1^{'}, \cdots, X_m^{'}] = [\underbrace{1, 0, \cdots, 0}_{field_1}, \cdots, \underbrace{0, 1, \cdots, 0}_{field_m}] \quad (1)$$

where $X_m^{'}$ is one-hot encoding representation in the m-th field; m is the size of fields; X_i represents the i-th input instance with m field representations. Following the former works [10], [17], [21], we apply the embedding layer to transform each field representation into the low-dimensional and dense embedding vector. Finally, the embedding output $E_i = [e_1, e_2, \cdots, e_p, \cdots, e_m]$ represents m embedding vectors of one input instance X_i, where $e_p \in \mathbb{R}^d$ denotes the embedding vector of the p-th field, and d is the dimension of embedding layer.

B. Broad Attentive Graph Fusion Layer

1) Attentive Graph Fusion Module: In this module, the embedding instance E_i is defined as multi-field input features. Next, we express these features in the form of a graph, that is $G = (N, \xi)$. Each node $n_p \in N$ in the graph represents a feature field p with embedding vector e_p, and the number of nodes $|N|$ is m. The edge ξ reflects intricate interactions between feature fields, and the edge weights denote the importance of different feature interactions. After the definition of graph structure, the module will optimize the adjacency matrix of edge weights, design a new bilinear-cross aggregation function, learn the aggregation impact of neighborhood nodes,



(a) first-order aggregation (b) second-order aggregation

Fig. 2: The architectures of different aggregation operations. In order to fully extract the cross features, a new bilinear-cross aggregation function integrates two different aggregation operations: (a) first-order aggregation; (b) second-order aggregation. The former is to enhance the affinity between the current node and its neighborhood nodes. The latter emphasizes building a bridge between two nodes without a direct connection.

and construct multi-hop feature fusion steps. These operations can effectively strengthen high-order feature representation and finally yield higher-quality cross features. Concretely, each node must qualify the impact of its neighbors and aggregate their cross information to output high-quality feature fusion of sophisticated node interactions. On the other hand, the steps of feature fusion can be further connected to generate high-order feature representation, called multi-hop feature fusion steps, i.e., the input of the latter step comes from the output of the former step. Formally, $H^l = f(\kappa^l, A, W^l)$ represents the node interactions representation results at feature fusion step l, where $f(\cdot)$ is the aggregation function; $A \in \mathbb{R}^{m \times m}$ is the adjacency matrix with edge weights, where m is the number of nodes; κ^l is the input features at step l, which needs to execute the element-wise addition with multi-field input features E_i; W^l denotes the trainable matrix of network weights. Then H^l and κ^l are entered into GRU (Gate Recurrent Unit), which can generate the input features κ^{l+1} at the next step $l+1$. At last, this module can yield the final result κ^L of feature representation, where L is the number of multi-hop feature fusion steps. Apparently, the adjacency matrix A and the aggregation function $f(\cdot)$ are fundamental to the effectiveness of node interactions in this module.

Attentional Edge Weights. Generally, different graph-based models deploy a variety of strategies to construct the adjacency matrix of edge weights. For instance, the conventional GNN model [32] usually adopts 0 and 1 to reflect the relationship between nodes. This way is too simple and rough to generate the accurate adjacency matrix for sophisticated node interactions. Thus Fi-GNN [13] utilizes the attention mechanism to explicitly learn the edge weights as follows:

$$\xi\left(n_p, n_q\right) = \frac{\exp\left(\text{ LeakyReLu } \left(W_\xi \left[e_p \| e_q\right]\right)\right)}{\sum_{k \in \mathcal{N}_p} \exp\left(\text{ LeakyReLu } \left(W_\xi \left[e_p \| e_k\right]\right)\right)} \tag{2}$$

where $\xi(n_p, n_q)$ denotes the edge weight between n_p and n_q; e_p and e_q are two embedding vectors of n_p and n_q; $\mathcal{N}_p$ is the neighborhood nodes of n_p; $||$ is the concatenation operation; W_ξ is the trainable matrix of attentive weights; $LeakyReLu$ (Leaky Rectified Linear Unit) is the non-saturated activation function. Recently, some state-of-the-art research work [6], [17] has proved that the product operation is better than the concatenate operation for multi-field feature interactions. Therefore, different from Fi-GNN [13], we propose a more fine-grained method to represent the edge weights as follows:

$$\xi\left(n_p, n_q\right) = \frac{\exp\left(\text{ ReLu } \left(\varrho \odot \left(\tilde{e}_p \odot \tilde{e}_q\right)\right)\right)}{\sum_{k \in \mathcal{N}_p} \exp\left(\text{ ReLu } \left(\varrho \odot \left(\tilde{e}_p \odot \tilde{e}_k\right)\right)\right)} \tag{3}$$

where $\odot$ denotes the Hadamard product operation that is element-wise multiply, for example, $[c_1, c_2, c_3] \odot [d_1, d_2, d_3] = [c_1 d_1, c_2 d_2, c_3 d_3]$; ϱ is the trainable vector of attentive weights; $\tilde{e}_p$ and $\tilde{e}_q$ are the normalized vectors of e_p and e_q; $ReLu$ (Rectified Linear Unit) is a new activation function, which is used to enhance the sparsity of feature selection instead of $LeakyReLu$; Accordingly, the adjacency matrix A can be defined as:

$$A\left(n_p, n_q\right) = \begin{cases} \xi\left(n_p, n_q\right), q \neq p \\ 0, \text{ otherwise} \end{cases} \tag{4}$$

where $A(n_p, n_q)$ denotes the edge weight from node n_p to its neighborhood node n_q. In brief, we construct a more fine-grained representation of attentional edge weights to accurately reflect the importance of node interactions.

Bilinear-Cross Aggregation. With the adjacency matrix A and the input features κ^l, the aggregation function $f(\cdot)$ will further update the state of each node by aggregating the cross information between itself and its neighborhood nodes. Generally, the traditional aggregation operation in GNN [32] is the simple multiply of adjacency matrix and node information, which is difficult to fully extract the cross features. Inspired by some graph-based researches [43], [44], we propose a new bilinear-cross aggregation function, which integrates two different aggregation operations: first-order aggregation and second-order aggregation. The former is to enhance the affinity between the current node and its neighborhood nodes. The latter emphasizes building a bridge between two nodes without a direct connection. As shown in Fig. 2(a), the first-order aggregation operation $\psi_p^{(1)}$ can update the state of the current

node n_p by connecting with its first-order neighborhood nodes $\mathcal{N}_p^{(1)}$, which is defined as:

$$\psi_p^{(1)} = \sum_{n_q \in \mathcal{N}_p^{(1)}} A(n_p, n_q)\kappa_p^l \odot \kappa_p^l \tag{5}$$

where $A(n_p, n_q)$ denotes the edge weight between n_p and n_q; κ_p^l denotes the input features of n_p at step l. $\odot$ denotes the Hadamard product operation. Besides the multiply of the adjacency matrix and input features, the first-order aggregation adds the product operation with the input features, which is beneficial to enhance the affinity between nodes. As shown in Fig. 2(b), unlike the first-order aggregation, the second-order aggregation operation $\psi_p^{(2)}$ considers the current node n_p to connect its second-order neighborhood nodes $\mathcal{N}_p^{(2)}$, which is defined as:

$$\psi_p^{(2)} = \sum_{n_u \in \mathcal{N}_p^{(2)}} (A(n_q, n_u)+I)((A(n_p, n_q)+I)\kappa_p^l \odot \kappa_p^l) \tag{6}$$

where $n_u \in \mathcal{N}_p^{(2)}$ denotes the second-order neighborhood node of n_p, i.e., $n_u \in \mathcal{N}_q^{(1)}, n_q \in \mathcal{N}_p^{(1)}$; $A(n_q, n_u)$ denote the edge weight between n_q and n_u; I denotes the unit matrix. The second-order aggregation not only adopts two-hop propagation to enlarge the aggregation range, but also reflects the interaction effect of the current node itself using the unit matrix. Finally, we integrate the first-order and second-order aggregations into a new bilinear-cross aggregation function to improve the representation of node interactions with different cross orders as follows:

$$G_p^l = W_\psi^l[\psi_p^{(1)} \oplus \psi_p^{(2)}, \psi_p^{(1)} \odot \psi_p^{(2)}] \tag{7}$$

where G_p^l denotes the bilinear-cross aggregation result for n_p; W_ψ^l denotes the trainable matrix of node aggregation weights; $\oplus$ and $\odot$ are the element-wise addition and the Hadamard product respectively.

Meanwhile, we apply the self-attention mechanism to get the aggregation impact from the neighborhood nodes to the current node. Thus we can define the attention weight vector φ_p^l of n_p to generate the representation result $H_p^l = G_p^l \odot \varphi_p^l$ of node interactions at the feature fusion step l, which can be calculated as follows:

$$\varphi_p^l = \frac{\exp\left(W_2^l \text{ReLu}(W_1^l[G_p^l \| k_p^l]) \odot k_p^l\right)}{\sum_{k \in \mathcal{N}_p} \exp\left(W_2^l \text{ReLu}(W_1^l[G_k^l \| k_p^l]) \odot k_p^l\right)} \tag{8}$$

where G_p^l and G_k^l denote the bilinear-cross aggregation results for n_p and its neighbor n_k respectively; W_1^l and W_2^l denote two trainable matrices of node attention weights; κ_p^l denotes the input features of n_p at step l; $\mathcal{N}_p$ is the neighborhood nodes of n_p; $||$ is the concatenation operation; $\odot$ is the Hadamard product; $ReLu$ is the activation function.

Finally, we introduce the GRU module to obtain the updated representation k_p^{l+1} of n_p at the next step $l+1$, which is calculated as follows:

$$\begin{aligned}
z_p^{l+1} &= sigmoid\left(W_z^l H_p^l + U_z^l \kappa_p^l + b_z^l\right), \\
r_p^{l+1} &= sigmoid\left(W_r^l H_p^l + U_r^l \kappa_p^l + b_r^l\right), \\
\tilde{\kappa}_p^{l+1} &= tanh\left(W_h^l H_p^l + U_h^l\left(r_p^{l+1} \odot \kappa_p^l\right) + b_h^l\right), \\
\kappa_p^{l+1} &= \tilde{\kappa}_p^{l+1} \odot z_p^{l+1} + H_p^l \odot \left(1 - z_p^{l+1}\right)
\end{aligned} \tag{9}$$

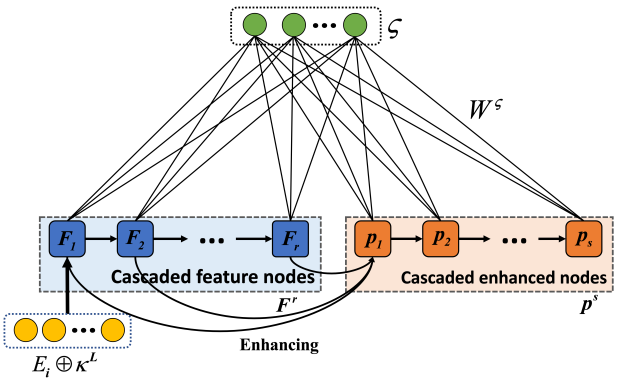


Fig. 3: The architecture of Broad Attention. The broad attention mechanism can construct the cascaded feature mapping to generate the feature nodes and the enhanced nodes, then calculate the attentive weights of global-aware nodes to reflect the importance of each cross feature.

where W_z^l, U_z^l, b_z^l, W_r^l, U_r^l, b_r^l, W_h^l, U_h^l, b_h^l are the trainable parameters of the GRU module; $sigmoid$ and $tanh$ are two activation functions. According to a series of operations for n_p, we can update the representations of all nodes to yield the input features κ^{l+1} at the next step $l+1$. After L propagation steps, we will output the final result κ^L of high-order feature representation in this module.

2) Broad Attentive Cross Module: The attentive graph fusion module learns the node interactions at the vector-wise level where all feature representations of cross nodes share the same weights, which can not specify the significance of each interaction at the granularity of feature dimensions. Considering this drawback, we propose a broad attentive cross module to efficiently refine high-order feature interactions at the bit-wise level. The new module is designed to dynamically increase the weights of important nodes and decrease the weights of uninformative nodes.

Generally, feature nodes are generated independently and have no relationship with each other by feature mapping in traditional BLS [16]. Unfortunately, the separated feature mapping will reduce the intrinsic correlation of input data. In this paper, we consider modeling feature interactions is to capture cross features among multi-field data with complex intrinsic correlation. Therefore, inspired by time-delayed neural networks [42], [45]–[47], we reconstruct a cascade BLS in the form of recurrent, which can strengthen the correlation between nodes. As shown in Fig. 3, we design a new broad attention mechanism based on the idea of cascading to construct the feature nodes and the enhanced nodes, strengthening the correlation between nodes to improve the performance of feature mapping. The cascaded feature mapping can be defined as follows:

$$\boldsymbol{F}_{z_i} = \phi^{z_i}(\phi^{z_i-1}(F_{z_i-1}W_{e_{j-1}} + \beta_{e_{j-1}}))W_{e_j} + \beta_{e_j} \tag{10}$$

where F_{z_i} is the z_i-th mapping feature nodes, $F_0 = \kappa^L \oplus E_i$, $z_i \in [1, r]$, r is the number of feature mapping; W_{e_j} and β_{e_j} are randomly generated weights and bias; $\oplus$ denotes the element-wise addition; $\phi^{z_i}(\cdot)$ denotes the $LeakyReLu$

activation function. All mapping features are concatenated to generate the final feature nodes $F^r \triangleq [F_1, \cdots, F_r]$. Next we execute the enhanced feature mapping as follows:

$$\boldsymbol{P}_{f_i} = \phi^{f_i}(\phi^{f_i-1}(P_{f_i-1}W_{h_{j-1}} + \beta_{h_{j-1}}))W_{h_j} + \beta_{h_j} \quad (11)$$

where P_{f_i} is the f_i-th enhanced feature node, $P_0 = F^r$, $f_i \in [1, s]$, s is the number of enhanced feature mapping; W_{h_j} and β_{h_j} are randomly generated weights and bias; $\phi^{f_i}(\cdot)$ denotes the $LeakyReLu$ activation function. All enhanced mapping features are concatenated to generate the final enhanced nodes $P^s \triangleq [P_1, \cdots, P_s]$.

After obtaining the feature nodes F^r and the enhanced nodes P^s, we can calculate the broad attentive weights ζ of global-aware nodes, reflecting the importance of each cross feature:

$$\begin{aligned} \zeta &= [F^r \| P^s] W^\zeta, \\ W^\zeta &= [F^r \| P^s]^+ \zeta = A^+ \zeta, \\ A^+ &= \lim_{\lambda \to 0} \left(\lambda I + A^T A\right)^{-1} A^T \end{aligned} \quad (12)$$

where $||$ is the concatenation operation; W^ζ is the trainable matrix of connecting weights; A^+ is the pseudoinverse of $A = [F^r | P^s]$; λ is an adjustment parameter with an initial value of 0.1; I is the unit matrix.

Finally, the representation Γ of cross features can be calculated as follows:

$$\Gamma = (W_2^b(\kappa^L \otimes E_i)) \odot (W_1^b(\kappa^L \oplus E_i)) \odot \sigma(pooling(\zeta)) \quad (13)$$

where W_1^b and W_2^b are two trainable matrices of network weights; ζ denotes the broad attentive weights of global-aware nodes; $\sigma(\cdot)$ denotes the $sigmoid$ activation function; $\odot$ denotes the Hadamard product; $\otimes$ denotes the Inner product; $\oplus$ denotes the element-wise addition. In this module, the broad attentive weights reflect the significance of node interactions. The high-order feature interactions are conducted at the bit-wise level, which can achieve the more fine-grained representation at each dimension of cross features.

C. Prediction Layer

After obtaining the high-quality cross features by the above layer, we can compute the final predictive result of our BaGFN model for multi-filed sparse data, which is defined as follows:

$$\hat{y} = sigmoid(w + \sum_{i=1}^{|\Gamma|} \Gamma_i) \quad (14)$$

where $\hat{y} \in [0, 1]$ is the final prediction probability; $sigmoid$ is an activation function; Γ is the final representation of feature interactions; w is a bias parameter; $|\Gamma|$ is the dimension of cross features Γ.

In the process of training, our BaGFN model takes Log-likelihood Loss as the loss function and then minimizes the objective function $\mathcal{L}$, which is defined as:

$$\mathcal{L} = -\frac{1}{M} \sum_{i=1}^{M} \left(y_i \log\left(\hat{y}_i\right) + (1 - y_i) \log\left(1 - \hat{y}_i\right)\right) \quad (15)$$

where M is the size of the training samples; y_i is the predictive ground truth of the i-th training instance; $\hat{y}_i$ is the predictive result of our new network. Meanwhile, we apply Back Propagation to continually update the trainable weight matrices of the above layers until the minimum value of loss is yielded. The overall training process of our BaGFN model is illustrated in Algorithm 1.

Algorithm 1 Training of our BaGFN model.

Input: All training samples $(X_i, y_i), i \in [1, M]$, X_i and y_i are the encoding input and the predictive ground truth of the i-th training instance respectively, M is the size of the training samples

Output: The trained network weights W

```
Set the batch size bsize and the iteration number IN;
Initialize all network weights W_0;
for t = 1; t ≤ IN do
    Execute a batch of training instances in parallel.
    Obtain the embedding output E_i from X_i;
    Construct the adjacency matrix A from E_i;
    for l ← 1 to L do
        for each node p do
            Compute the bilinear-cross aggregation G_p^l;
            Compute the self-attention weight vector φ_p^l;
            Get the representation H_p^l from (G_p^l, φ_p^l);
            Use the GRU module to obtain κ_p^{l+1};
        end for
        Generate κ^{l+1} after updating all nodes;
    end for
    Generate κ^L after L propagation steps;
    for z_i ← 1 to r do
        Calculate each feature node F_{z_i} from (κ^L, E_i);
    end for
    Obtain all feature nodes F^r;
    for f_i ← 1 to s do
        Calculate each enhanced node P_{f_i} from F^r;
    end for
    Obtain all enhanced nodes P^s;
    Construct the broad attentive weights ζ from (F^r, P^s);
    Calculate the cross features Γ from (κ^L, E_i, ζ);
    Obtain the final predictive result ŷ from Γ;
    Minimize the objective function L from (ŷ, y);
    Update W_t using Back Propagation for L ↓;
end for
return W ← W_IN;
```

D. Complexity Analysis

Referring to the computation mode of [11], the time consumption of computing an instance in Algorithm 1 mainly comes from four operations: calculating embedding output, constructing adjacency matrix, iteratively updating graph nodes, and computing broad attention. Firstly, since the one-hot encoding representation of each field is transformed into a fixed low-dimensional embedding vector, the time complexity of calculating embedding output is represented as $O(Cd)$, where C is the total length of all one-hot encoding representations and d is the dimension of embedding space $\mathbb{R}^d$. Secondly, the time complexity of constructing an $m \times m$ adjacency matrix

is $O(m^2d)$, where m is the number of fields and also the number of graph nodes. Thirdly, when updating a graph node, the time complexity of bilinear-cross aggregation, including first-order aggregation and second-order aggregation, is defined as $O(md + m^2d)$. Since the self-attention and GRU operations will transform each other between $\mathbb{R}^d$ and a new space $\mathbb{R}^{d'}$, their time complexity is represented as $O(mdd')$. Thus when iteratively updating all graph nodes, the time complexity can be computed as $O(Lm(md + m^2d + mdd'))$, where L is the number of propagation steps. Fourthly, since the broad attention module needs to carry out the cross calculation between nodes, its time complexity is expressed as $O(mn^2d^2)$, where n is the number of nodes in cascade BLS. Finally, the time complexity of computing an instance is defined as $O(Cd+m^2d+Lm^2d+Lm^3d+Lm^2dd'+mn^2d^2)$, which can be simplified to $O(C+m^2(dd'+md)+mn^2d^2)$ because d and L are usually small, and C is usually large and $C \gg d$. When training all instances in parallel, the final time complexity of our proposed method is $O(M'(C+m^2(dd'+md)+mn^2d^2))$, $M' = M \times IN/bsize$, where M is the size of training instances, IN is the number of training iteration, and $bsize$ is the size of a batch.

According to the definition of [13], the space complexity of our proposed method is a measure of the total amount of parameters, which consist of trainable matrices of several key parts, including the embedding layer, the attentive graph fusion module, and the broad attentive cross module. Firstly, the average size of embedding matrices is defined as $c_1 \times d$, where c_1 is the average length of all one-hot encoding representations. Thus the space complexity of the embedding layer is $O(mc_1d)$. Secondly, since the attentive graph fusion module will transform each other between $\mathbb{R}^d$ and a new space $\mathbb{R}^{d'}$, the size of each trainable matrix can be defined as $d \times d'$. In an iteration step, the size of all trainable matrices is $c_2 \times d \times d'$, where c_2 is the number of matrices. Thus the space complexity of the attentive graph fusion module can be defined as $O(Lc_2dd')$. Thirdly, the size of the trainable broad attention matrix is $c_3d \times c_3d$, where c_3 is the number of nodes in cascade BLS. Thus the space complexity of the broad attentive cross module is expressed as $O({c_3}^2d^2)$. Finally, since the parameters of these trainable matrices are shared, the final space complexity is defined as $O(c_1md + c_2Ldd' + {c_3}^2d^2)$, which is fixed during the training of our BaGFN model.

IV. Experimental Results

In this section, we conduct extensive experiments to answer the following questions:

- **RQ 1** How does our model perform in learning cross features compared with the state-of-the-art models on multi-field sparse data? Is it efficient for model training?
- **RQ 2** How do the different combinations of aggregation function types (first-order, second-order, bilinear-cross and bilinear-concatenate) and edge weights (concatenate and product operations) in attentive graph fusion module impact its performance?
- **RQ 3** How do the different settings of the network influence the performance of our model? Can our model reflect the significance of different feature fields?
- **RQ 4** Similar to the state-of-the-art models, does our model also further enhance the performance by integrating implicit feature interactions?

We will deal with these questions after presenting several fundamental experimental configurations.

TABLE II: Statistics of three evaluation datasets.

Dataset	Instances	Fields	Features (Sparse)
Avazu	40,428,967	23	1,544,488
Criteo	45,840,617	39	998,960
Tobacco	166,626	249	146,103

A. Experimental Configuration

1) Datasets: In order to evaluate the validity of our BaGFN model, we conduct a series of experiments on two well-known public benchmark datasets and a new industrial dataset. The statistics of the datasets are summarized in Table II.

Avazu: The Avazu dataset consists of users' click records of several days on mobile advertisements with 40 million of data instances. This dataset is initially used for Kaggle CTR prediction competition and later widely used in the evaluation benchmark for many CTR prediction models. For each click instance, it has 23 fields spanning from user/device features to ad attributes.

Criteo: Criteo is a famous real-world display ad dataset with each ad display information and corresponding user click logs. This dataset is widely used in the evaluation of various CTR prediction models. It contains users' click logs with 45 million samples, and each sample has 39 fields, including 13 continuous numerical fields and 26 categorical fields.

Tobacco: The new dataset consists of more than 160,000 inspection records for 40,000 tobacco stores from 2014 to 2018. Similar to CTR prediction, this dataset is to predict illegal sales in the real-world tobacco industry. Each sample contains 249 fields (195 dynamic numerical fields and 54 static categorical fields) from individual information, dynamic orders, and sales log, and has numerous null values and a few anomaly data.

Data Preparation: For Avazu and Criteo datasets, we remove the infrequent values appearing less than 10, 5 times, respectively, and treat them as a single value $< unknown >$. Since the numerical fields may have large variance and then hurt the model performance, we normalize numerical values by transforming a value z to $\log_2(z)$ if $z > 2$ which is first proposed by the winner of Criteo Competition[3]. For the Tobacco dataset, we replace null value with -1 and normalize numerical values to eliminate the quantum influence between different ranges of numerical fields. For three datasets, we randomly select 80% of all samples for training and randomly divide the rest into equal validation sets and test sets. In this paper, all methods follow this unified way of data preparation for a fair comparison.

[3]https://www.csie.ntu.edu.tw/~r01922136/kaggle-2014-criteo.pdf

2) Evaluation Metrics: So as to evaluate the prediction performance of our BaGFN model on Avazu, Criteo, and Tobacco datasets, we adopt two popular metrics:

AUC (Area Under the ROC Curve) measures the probability that a predictive model will allocate a higher score to a positive instance than a randomly chosen negative instance. AUC's upper bound is 1, and the larger value indicates a better performance.

Logloss (Cross Entropy) is the cross-entropy loss to assess the performance of a classification model, measuring the distance between the predicted score and the true label for each sample. The smaller Logloss value denotes better performance.

3) Comparison Methods: We compare our BaGFN model with three types of the existing methods: (i) Modeling first-order interactions that linearly sum up raw features; (ii) FM-based methods that learn second-order cross features; and (iii) Deep learning based models that model high-order feature interactions. We briefly describe these methods as follows:

(i) **LR (Logistic Regression)** [48] captures first-order cross features via summing up raw individual features linearly.

(ii) **FM** [6] is a benchmark factorization model, which models the second-order interactions between paired features from the inner product of vectors.

(iii) **AFM** [20] is an extension of FM, which learns the significance of different second-order cross features with the help of the attention mechanism.

(iv) **CrossNet (Deep&Cross)** [2] attempts to model feature interactions in an explicit manner by taking the outer product of feature vectors at the bit-wise level.

(v) **DeepCrossing** [49] integrates the advantages of deep fully-connected neural networks and residual connections to capture non-linear high-order feature interactions via an implicit manner.

(vi) **NFM** [8] feeds the element-wise product of feature vectors into a bi-interaction pooling layer to model the second-order interactions, and then adopts deep fully-connected neural networks to capture non-linear high-order interactions in an implicit fashion.

(vii) **PNN** [17] concatenates pairwise inner or outer products of input feature vectors to learn high-order feature interactions.

(viii) **CIN** is the core of xDeepFM [10], which produces high-order cross features by computing the outer product of input feature vectors and then compressing the feature maps originated from the outer product to renew the representation of each feature vector.

(ix) **HOFM (High-Order FM)** [50] is a high-order version of FM, which proposes efficient kernel-based algorithms for modeling non-linear high-order factorization machines.

(x) **AutoInt** [11] utilizes a multi-head self-attentive neural network with residual connections to model the high-order feature interactions in an explicit manner.

(xi) **HoAFM** [21] designs a cross interaction layer and bit-wise attention mechanism to account for the high-order sparse feature interactions in an explicit manner.

(xii) **Fi-GNN** [13] generates sophisticated feature interactions by designing a novel graph neural network to model the graph-structured features in an explicit manner.

(xiii) **DCN-M*** is an updated version of DCN-M [18] without implicit feature interactions, which improves the ability of CrossNet [2] to make it more practical in large-scale industrial settings.

(xiv) **DIFM*** is an updated version of DIFM [22] without implicit feature interactions, which integrates various components and reweights the original feature representations at the bit-wise and vector-wise levels simultaneously.

4) Implementation Details: In this paper, all experiments are conducted over a sever equipped with 2 NVIDIA Titan XP GPUs. Here, we refer to the default or optimal parameter configuration of these state-of-the-art models. For DeepCrossing, we employ four feed-forward layers, each with 100 hidden units. For NFM, we adopt one hidden layer with size 200 on top of the Bi-Interaction layer. For HoAFM, we set its depth to 3. For CrossNet and CIN, we employ three interaction layers. For our BaGFN model, we mainly consider the configuration of five parameters, including embedding dimension, epochs, iteration steps, batch size, and learning rate. For three different evaluation datasets, we conduct a series of comparison experiments to find the best parameters settings and yield the optimal prediction results. Moreover, we apply the Adam optimizer to optimize all DNN-based models.

*B. Performance Comparison (**RQ1**)*

According to the above experimental configurations, we compare our BaGFN model with all kinds of individual models, including first-order, second-order, and high-order feature interactions. As illustrated in Table III and Fig. 4, we summarize the performance of these models and obtain the following observations:

(1) The linear model LR fails to yield a good performance on the three datasets compared to the other nonlinear models, suggesting that modeling nonlinear relations among features is essential to improve the accuracy of CTR prediction.

(2) FM and AFM, which capture second-order nonlinear feature interactions, achieve better predictive results as compared to LR, showing the pairwise interactions via the inner product of feature vectors have a positive impact on prediction performance. Moreover, AFM generates better predictive results than FM on all three datasets, which indicates that the attention on different feature interactions endows the model with better interaction ability.

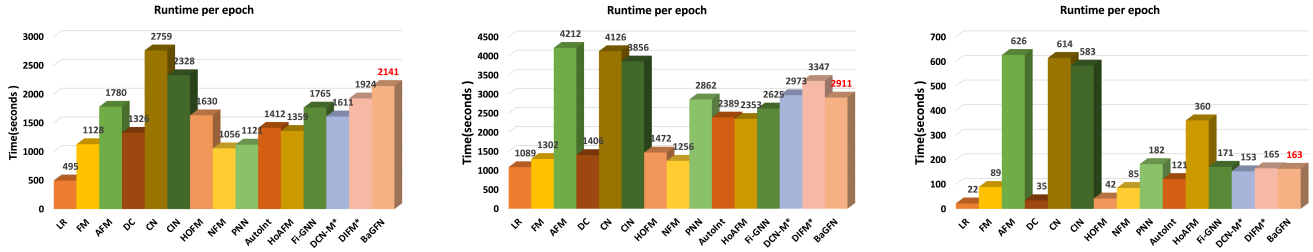
(3) Most high-order models substantially outperform those of learning low-order feature interactions on the three evaluation datasets. This is consistent with the intuition that modeling high-order interactions among features is more significant for prediction performance.

(4) Fi-GNN exhibits better prediction performance compared to the other baseline models, indicating that feature interactions based on graph neural networks have the stronger capability of feature representation.

(5) Our BaGFN model yields the best performance on all three datasets and achieves innovative improvements over the state-of-the-art models. BaGFN has the highest AUC values (0.7803, 0.8094, and 0.8870) and the lowest Logloss values (0.3794, 0.4425, and 0.3329) on Avazu, Criteo, and Tobacco.

TABLE III: The overall performances of various models on Avazu, Criteo, and Tobacco datasets.

Model Category	Model	Avazu			Criteo			Tobacco		
		AUC	Logloss	p-value	AUC	Logloss	p-value	AUC	Logloss	p-value
First-Order	LR [48]	0.7558	0.3963	0.00031	0.7822	0.4696	0.00022	0.7243	0.4493	0.00019
Second-Order	FM [6]	0.7710	0.3855	0.00058	0.7833	0.4695	0.00085	0.7883	0.4111	0.00011
	AFM [20]	0.7716	0.3852	0.00067	0.7941	0.4581	0.00047	0.7958	0.4092	0.00019
High-Order	NFM [8]	0.7711	0.3863	0.00018	0.7959	0.4563	0.00062	0.7926	0.4083	0.00059
	DeepCrossing [49]	0.7648	0.3886	0.00075	0.8010	0.4512	0.00022	0.8698	0.3561	0.00063
	CrossNet [2]	0.7671	0.3867	0.00021	0.7908	0.4589	0.00011	0.8618	0.3588	0.00082
	PNN [17]	0.7753	0.3835	0.00028	0.7998	0.4521	0.00010	0.8736	0.3461	0.00051
	HOFM [50]	0.7703	0.3853	0.00011	0.8008	0.4506	0.00071	0.8755	0.3473	0.00034
	CIN [10]	0.7759	0.3829	0.00045	0.8011	0.4515	0.00063	0.8801	0.3398	0.00071
	AutoInt [11]	0.7755	0.3822	0.00099	0.8063	0.4453	0.00061	0.8369	0.3951	0.00069
	HoAFM [21]	0.7761	0.3820	0.00090	0.8057	0.4468	0.00031	0.8839	0.3395	0.00059
	Fi-GNN [13]	0.7762	0.3825	0.00035	0.8062	0.4453	0.00077	0.8843	0.3353	0.00065
	DCN-M* [18]	0.7649	0.3880	0.00068	0.7992	0.4514	0.00038	0.8820	0.3465	0.00039
	DIFM* [22]	0.7660	0.3874	0.00019	0.7905	0.4592	0.00059	0.8702	0.3544	0.00089
	BaGFN	**0.7803**	**0.3794**	—	**0.8094**	**0.4425**	—	**0.8870**	**0.3329**	—



(a) Avazu (b) Criteo (c) Tobacco

Fig. 4: The training runtimes of all models for each epoch on Avazu, Criteo, and Tobacco datasets. Note: The baseline models include LR [48], FM [6], AFM [20], DC represents DeepCrossing [49], CN represents CrossNet (Deep&Cross) [2], CIN is the core of xDeepFM [10], HOFM [50], NFM [8], PNN [17], AutoInt [11], HoAFM [21], Fi-GNN [13], DCN-M* [18], DIFM* [22]. Compared with the other models, our BaGFN model has a moderate time consumption for training.

The previous works [2], [9], [11], [13] have proved that the improvement with respect to AUC at 0.001-level is significant for the CTR prediction task. Our proposed model presents great superiority over those state-of-the-arts models, owing to the effectiveness of integrating graph structure and attention mechanism on modeling feature interactions.

(6) Compared with the other baseline models, the relative performance improvement of our BaGFN model is progressively higher on Avazu, Criteo, and Tobacco datasets, which may be because the number of feature fields is different across the three evaluation datasets, with the Tobacco dataset owning the most feature fields and Avazu dataset owning the least. Generally, more feature fields are beneficial to capture more various cross features and achieve better performance improvement on multi-field sparse dataset.

(7) For formal statistical analysis, Wilcoxon Test is introduced to calculate and compare the p-value, which indicates whether the differences in performance are statistically significant or not. With the prediction results of these models on Avazu, Criteo, and Tobacco datasets, we can calculate the p-values between our BaGFN model and the other models, and further compare these p-values with the significance level to analyze their statistically significant differences. As illustrated in Table III, all p-values on three evaluation datasets are less than 0.01, which demonstrates that there is an extremely significant difference between our BaGFN model and the other state-of-the-art model.

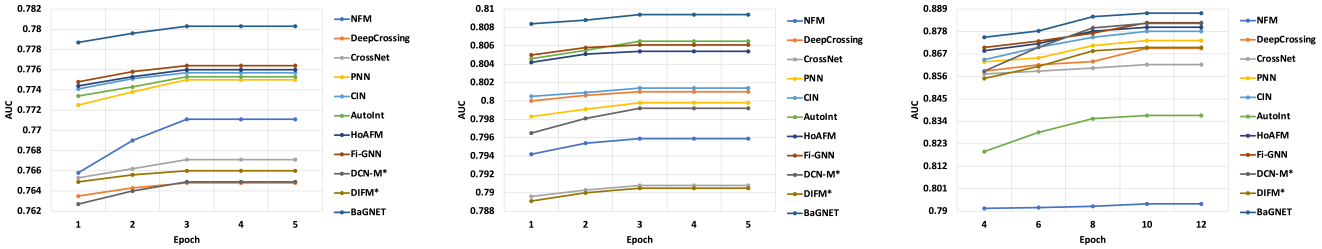
(8) Besides, we also analyze the running time in the process of model training. Clearly, LR becomes the most efficient model because of its simplicity. AFM, CrossNet, and CIN are the highest time-consuming models due to their complex architectures. The time complexity of our proposed method is $O(M'(C+m^2(dd'+md)+mn^2d^2))$, which indicates that the running time of training is related to the number of training instances, the total length of all one-hot representations, the number of graph nodes, and the embedding dimension. Since the number of fields and the embedding dimension are usually small, our BaGFN model will not add more training time than the other high-order models. Fig. 4 illustrates the time consumption of model training for one epoch on Avazu, Criteo, and Tobacco datasets. Compared with the other models, our BaGFN model has a moderate time consumption for training.

C. Comparison of Different Aggregation Functions and Edge Weights (*RQ2*)

In this section, we exploit several variants of our model to investigate how the different aggregation functions and edge weights in the attentive graph fusion module affect the prediction results. For different aggregation functions, we utilize the original aggregation as same as GCN [15], termed $model_{org}$, and also construct two variants only using the first-order

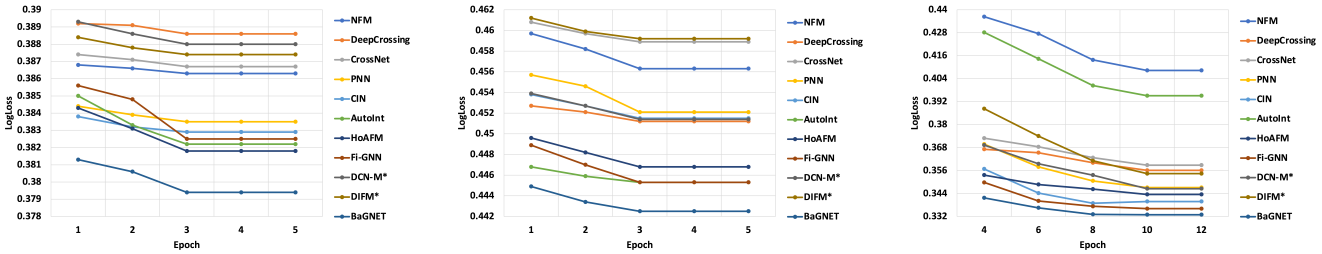
TABLE IV: The performances of different aggregation functions and edge weights in attentive graph fusion module.

Model	Avazu		Criteo		Tobacco	
	AUC	Logloss	AUC	Logloss	AUC	Logloss
$model_{org}$	0.7795	0.3798	0.8086	0.4431	0.8795	0.3408
$model_{first}$	0.7799	0.3798	0.8089	0.4428	0.8798	0.3345
$model_{second}$	0.7794	0.3797	0.8085	0.4430	0.8799	0.3344
$model_{concat}$	0.7800	0.3796	0.8090	0.4427	0.8834	0.3334
$model_{ec}$	0.7794	0.3799	0.8090	0.4426	0.8816	0.3341
$model_{final}$	**0.7803**	**0.3794**	**0.8094**	**0.4425**	**0.8870**	**0.3329**



(a) Avazu (b) Criteo (c) Tobacco

Fig. 5: The AUC performance of high-order models for different epochs on Avazu, Criteo, and Tobacco datasets. Note: The baseline models include DeepCrossing [49], CrossNet (Deep&Cross) [2], PNN [17], CIN is the core of xDeepFM [10], NFM [8], AutoInt [11], HoAFM [21], Fi-GNN [13], DCN-M* [18], DIFM* [22]. Compared with the other models, our BaGFN model has the highest AUC values under different epochs. The optimal epochs are respectively fixed to 3, 3, 10 on three evaluation datasets.



(a) Avazu (b) Criteo (c) Tobacco

Fig. 6: The Logloss performance of high-order models for different epochs on Avazu, Criteo, and Tobacco datasets. Note: The baseline models include DeepCrossing [49], CrossNet (Deep&Cross) [2], PNN [17], CIN is the core of xDeepFM [10], NFM [8], AutoInt [11], HoAFM [21], Fi-GNN [13], DCN-M* [18], DIFM* [22]. Compared with the other models, our BaGFN model has the lowest Logloss values under different epochs. The optimal epochs are respectively fixed to 3, 3, 10 on three evaluation datasets.

and second-order operations in Eq. (5) and Eq. (6), which are termed as $model_{first}$ and $model_{second}$ respectively. In addition, we define $model_{concat}$ to represent the concatenation operation of Eq. (5) and Eq. (6), and denote the bilinear-cross aggregation in Eq. (7) as $model_{final}$. Meanwhile, for computing edge weights, $model_{ec}$ represents the concatenation operation as same as Fi-GNN [13], and $model_{final}$ indicates the product operation in Eq. (3). The final experimental results are shown in Table IV with the following findings:

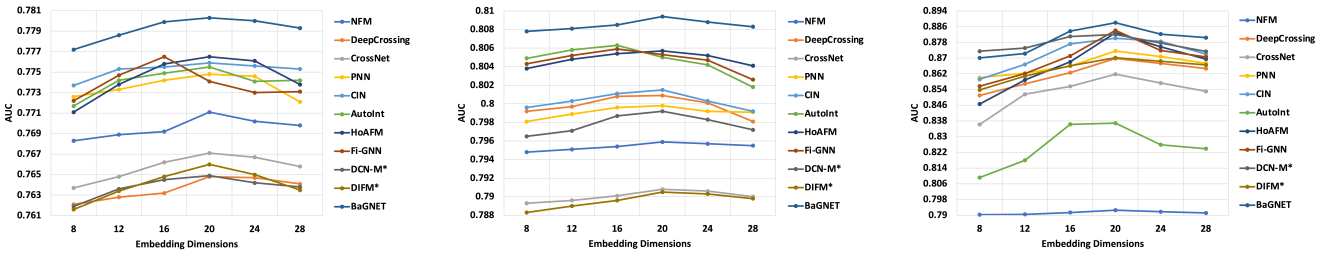
(1) $model_{first}$ outperforms $model_{org}$ on three evaluation datasets, which indicates that the first-order operation can execute more effective node interactions than the original sum operation.

(2) $model_{second}$ is not always better than $model_{first}$ and $model_{org}$, which indicates that the single second-order operation is unstable and needs to further combine the first-order operation for the stronger performance.

(3) Unlike $model_{first}$ and $model_{second}$, $model_{concat}$ obtains the better performance on three evaluation datasets, which shows that concatenation operation can further improve the ability of linear aggregation by integrating first-order and second-order operations.

(4) $model_{final}$ is superior to all types of aggregation operations, indicating that the bilinear-cross aggregation in our BaGFN model can significantly boost the prediction performance.

(5) Compared with $model_{ec}$, $model_{final}$ achieves a significant improvement on three evaluation datasets, which indicates that the product operation in our BaGFN model can effectively compute the edge weights and better represent the relationship



(a) Avazu (b) Criteo (c) Tobacco

Fig. 7: The AUC performance of high-order models for diverse embedding dimensions on Avazu, Criteo, and Tobacco datasets. Note: The baseline models include DeepCrossing [49], CrossNet (Deep&Cross) [2], PNN [17], CIN is the core of xDeepFM [10], NFM [8], AutoInt [11], HoAFM [21], Fi-GNN [13], DCN-M* [18], DIFM* [22]. Compared with the other models, our BaGFN model has the highest AUC values under different embedding dimensions. The optimal embedding dimensions are fixed to 20 on all three evaluation datasets.

between two nodes than the concatenation operation.

*D. Hyper-parameter Investigation (**RQ3**)*

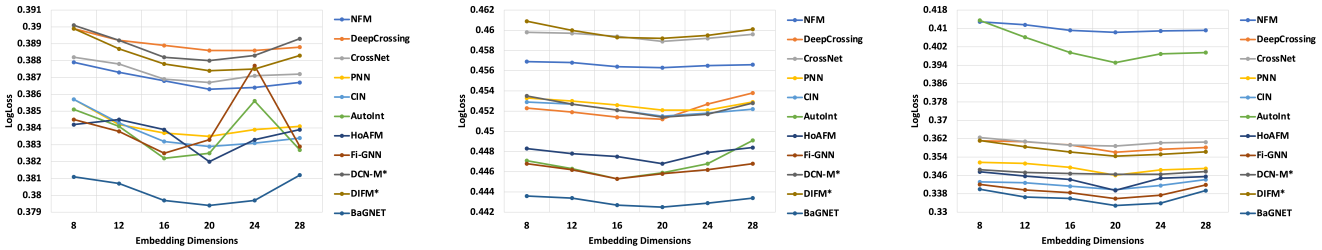
In this section, we will conduct several configuration investigations for our BaGFN model with different hyper-parameters. Specifically, we investigate the following crucial factors: (1) different epochs; (2) diverse embedding dimensions; (3) iteration steps of multi-hop feature fusion; (4) ablation study of attentional components; (5) broad attentive weights.

1) Different Epochs: Epoch is a crucial parameter for model training. Generally, an epoch refers to the process of sending all data into the network to complete a forward calculation and back propagation. During the training, it is not enough to update network weights by a single epoch. Thus we need multiple epochs to achieve the fitting and convergence of training. In this paper, Fig. 5 and Fig. 6 show the prediction performance of high-order models under different epochs during model training. From these figures, we can obtain some important results about the influence of different epochs. Firstly, our BaGFN model can obtain the best prediction performance when its epoch values are respectively fixed to 3, 3, and 10 on Avazu, Criteo, and Tobacco datasets. The main reason is that with the increase of epoch, the network weights of our BaGFN model are updated iteratively during training, and its performance is continuously improved until convergence. However, the larger epoch cannot always get the better performance because of the overfitting problem, and the optimal epoch values for different datasets are also different due to the convergence speed. Secondly, compared with the other models, our BaGFN model always possesses the highest AUC and lowest Logloss values under different epochs. The main reason is that our BaGFN model has the more excellent ability of modeling high-order feature interactions. The whole training of our BaGFN model is very stable, and effective because all AUC values are increasing and all Logloss values are decreasing on three evaluation datasets. Thirdly, the optimal epoch value on the Tobacco dataset is larger than those on the other two datasets. The main reason is that the number of epoch is related to the diversity of dataset. The Tobacco dataset contains 249 fields, much more than 23 fields on Avazu dataset and 39 fields on the Criteo dataset. The stronger the diversity is, the larger the epoch value should be. The high-order feature interactions on this dataset are more complex and need more epochs to achieve the convergence of training.

2) Diverse Embedding Dimensions: The Embedding dimension is of key significance to provide sufficient representation capability. Thus we explore how the performance of high-order models changes under the different embedding dimensions. As shown in Fig. 7 and Fig. 8, since the larger embedding dimensions yield more information, the performance of high-order models can continue to increase when the embedding dimension is below the optimal setting. However, when the embedding dimensions surpass the optimal setting, too much information might cause that the models are over-fitted, and their prediction performance will decrease. Compared with the other models, our BaGFN model can reach the best prediction performance on three evaluation datasets when all three embedding dimensions are set to 20, which is the optimal setting.

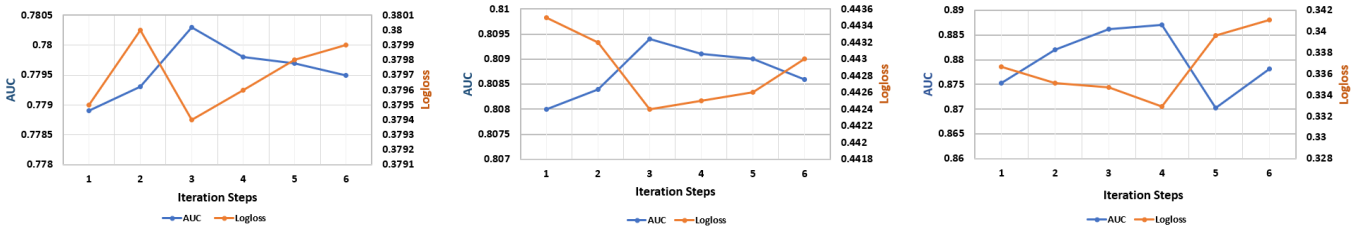
3) Iteration Steps of Multi-hop Feature Fusion: Generally, the iteration steps of multi-hop feature fusion are constructed to generate a high-order representation for feature interactions. In order to find out the influence of multi-hop feature fusion, we investigate the performance of our BaGFN model under the different iteration steps. As shown in Fig. 9, the moderate iteration steps can effectively improve the prediction performance on three evaluation datasets. For Avazu and Criteo datasets, our proposed model can reach the best performance when the iteration steps are set to 3. For the Tobacco dataset, the optimal iteration step is set to 4. Since Avazu and Criteo datasets possess 23 and 39 fields respectively, three iteration steps are sufficient to generate high-order feature interactions. In contrast, since the Tobacco dataset possesses 249 fields, much more than Avazu and Criteo datasets, more iteration steps are required to adequately interact with other nodes in the graph network.

4) Ablation Study of Attentional Components: Although we have yielded powerful empirical results for several attentional components in our BaGFN model, the results have



(a) Avazu (b) Criteo (c) Tobacco

Fig. 8: The Logloss performance of high-order models for diverse embedding dimensions on Avazu, Criteo, and Tobacco datasets. Note: The baseline models include DeepCrossing [49], CrossNet (Deep&Cross) [2], PNN [17], CIN is the core of xDeepFM [10], NFM [8], AutoInt [11], HoAFM [21], Fi-GNN [13], DCN-M* [18], DIFM* [22]. Compared with the other models, our BaGFN model has the lowest Logloss values under different embedding dimensions. The optimal embedding dimensions are fixed to 20 on all three evaluation datasets.



(a) Avazu (b) Criteo (c) Tobacco

Fig. 9: The performance of our BaGFN model under different iteration steps of multi-hop feature fusion. For Avazu and Criteo datasets, our proposed model can reach the best prediction performance when the iteration steps are set to 3; for Tobacco dataset, the optimal iteration step is set to 4.

TABLE V: Ablation study of the partial attentional components in our BaGFN model.

Datasets	Models	AUC	Logloss
Avazu	NO-BROAD	0.7796	0.3799
	NO-SA	0.7801	0.3795
	BASE	0.7803	0.3794
Criteo	NO-BROAD	0.8087	0.4429
	NO-SA	0.8088	0.4427
	BASE	0.8094	0.4425
Tobacco	NO-BROAD	0.8721	0.3367
	NO-SA	0.8715	0.3360
	BASE	0.8870	0.3329

not demonstrated their specific contributions. Therefore, we conduct ablation experiments over our proposed model to better understand their relative significance. We define our model with full attentional components as the BASE model and construct two incomplete models by removing the partial attentional components in the following manners:

(1) **No-BROAD**: remove the broad attentive cross module from the BASE model;

(2) **No-SA**: remove the self-attentive part in the attentive graph fusion module from the BASE model.

We compare the prediction results of the No-BROAD, No-SA, and BASE models on three evaluation datasets, and Table V illustrates the following observations:

(1) Compared with the BASE model, the performances of the No-BROAD and No-SA models drop apparently, which demonstrates that two partial attentional components are essential to improve the prediction performance.

(2) For the No-BROAD and No-SA models, much larger drops appear on the Tobacco dataset compared to the other datasets, which indicates that the attentional components play a greater role on the dataset with more feature fields.

5) Broad Attentive Weights: We design the broad attentive cross module for learning the weights of feature nodes, which can reflect the significance of feature fields on the overall prediction result. Fig. 10 shows the heat map of global-level and case-level broad attentive weights on the Avazu dataset, which indicates that the feature fields play different roles in the clicking behaviors. The global indicates a globally averaged one of all samples. The case1, case2, case3 and case4 are randomly selected from all samples, whose predictive scores are $[0.92, 0.15, 0.98, 0.99]$ and labels are $[1, 0, 1, 1]$ respectively. At the global level, we can find that the feature field app_category plays the strongest role in the clicking behavior; at the case level, we observe that in three of the cases, the feature field app_category has the largest influence on predicting the individual clicking behavior. It is reasonable since the Avazu dataset focuses on the mobile scene, where the app-related information is a critical factor for CTR prediction.

TABLE VI: The performances of various models after integrating implicit feature interactions.

Model	Avazu		Criteo		Tobacco	
	AUC	Logloss	AUC	Logloss	AUC	Logloss
Wide&Deep (LR) [4]	0.7752	0.3823	0.8028	0.4495	0.8562	0.3655
DeepFM (FM) [9]	0.7755	0.3831	0.8067	0.4448	0.8651	0.3614
Deep&Cross (CrossNet) [2]	0.7735	0.3834	0.8071	0.4449	0.8693	0.3591
xDeepFM (CIN) [10]	0.7771	0.3826	0.8074	0.4443	0.8835	0.3401
BaGFN$^+$	**0.7809**	**0.3792**	**0.8098**	**0.4421**	**0.8873**	**0.3319**

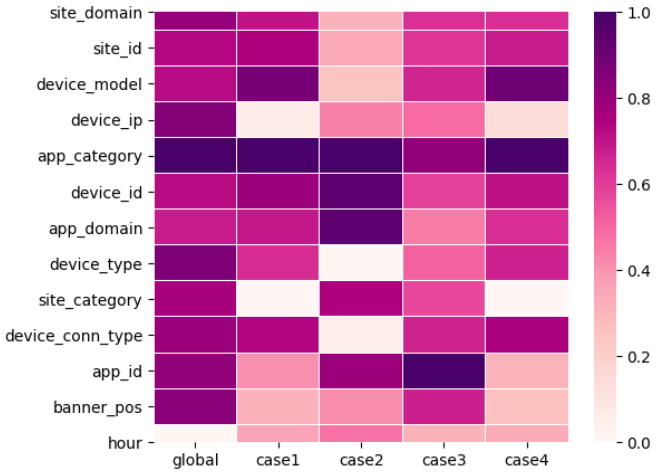


Fig. 10: Heat map of global-level and case-level broad attentive weights on the Avazu dataset, which demonstrates the significance of different feature fields on the final CTR prediction.

*E. Integrating Implicit Interactions (**RQ4**)*

Feed-forward neural networks are able to capture implicit feature interactions and have been extensively integrated into various CTR prediction models [2], [4], [9], [10]. To investigate whether integrating implicit feature interactions further enhances the prediction performance, we combine our BaGFN model with a three-layer feed-forward neural network via joint training. Our enhanced model is compared with the following state-of-the-art models:

(1) **Wide&Deep** [4] combines the outputs of logistic regression and feed-forward neural networks;

(2) **DeepFM** [9] feeds the output of an embedding layer to FM and feed-forward neural network, integrating the outputs of the two components to produce the final result;

(3) **Deep&Cross** [2] is the extension of CrossNet by introducing feed-forward neural networks;

(4) **xDeepFM** [10] is the extension of CIN by integrating feed-forward neural networks.

As shown in Table VI, based on the prediction performances on three evaluation datasets, we have the following observations:

(1) all models that integrate implicit interactions clearly boost their prediction ability, which demonstrates the effectiveness of feed-forward neural networks.

(2) our enhanced model with feed-forward neural networks outperforms all other models and achieves the new state-of-the-art performances on three evaluation datasets.

F. Discussion

Generally, the final model prediction accuracy depends on the quality of modeling feature interactions among multi-field sparse data. Our BaGFN model constructs the attentive graph fusion module to aggregate graph node information and strengthen high-order feature representation, further the broad attentive cross module is designed to learn the significance of different interactions and refine high-order feature interactions at the bit-wise level. After a series of experimental comparisons and analyses, our BaGFN model has three main advantages compared with the state-of-the-art methods: (a) the fine-grained cross features are extracted in an explicit manner, which can enhance the model explanations and get rid of negative and uncontrollable feature interactions derived from a rather implicit manner; (b) the multi-hop feature fusion with bilinear-cross aggregation is more comprehensive and more flexible for sophisticated high-order interactions; (c) the attentional components explore the importance of different cross features, which can identify the potential concerns of feature interactions more effectively. On the other hand, since the way of feature interaction is more complex, our BaGFN model also has some limitations: (a) more model parameters need to be trained, which consumes more computing resources; (b) for different datasets, manual hyper-parameter adjustment is time-consuming and laborious; (c) multi-hop feature fusion is easy to suffer from overfitting in model training due to inappropriate iteration steps. In brief, we have conducted a series of experiments to answer four key questions (**RQ1**-**RQ4**), and the final responses to these questions have also demonstrated the advantages and disadvantages of our BaGFN model.

V. Conclusion

In this paper, we establish a novel network named BaGFN as an abbreviation for Broad Attentive Graph Fusion Network, which aims to integrate graph structure and attention mechanism to generate powerful feature representation and model sophisticated high-order feature interactions among multi-field sparse data. Besides the general embedding layer and prediction layer, the broad attentive graph fusion layer is the core of our BaGFN model. On the one hand, we design a novel attentive graph fusion module in this layer to aggregate graph node information and generate high-order feature representation. On the other hand, we construct a new broad attentive cross module in this layer to highlight the different importance of cross features and refine high-order feature interactions at the bit-wise level. Finally, a variety of experimental results on three evaluation datasets demonstrate the excellent performance of our proposed model.

In the future, there are three main directions for better modeling feature interactions. Firstly, our graph neural network currently focuses on homogeneous graphs with a single node and edge type, i.e., multi-field data is defined as a specific type, which usually only requires aggregating a single type of neighbors to update node representation. Next, we will consider heterogeneous graphs with multiple nodes and edge types for more elaborate modeling of feature interactions. Secondly, temporal prediction analysis inevitably involves features that are dynamically changing over time. However, the current graph neural network is a static graph mode and lacks full consideration for how efficiently to handle dynamic feature fields. Therefore, we will introduce dynamic graphs for real-time feature change to immediately update node representation of time evolution patterns. Lastly, with the success of the Transformer model in machine learning applications, we will also try to integrate the Broad Attention module in our BaGFN model into the Transformer model like [51] so that the feature interactions can be captured more efficiently. Besides, the vector of locally aggregated descriptors (VLAD) has been a new trend and inspired by [52], we will also attempt to further optimize VLAD to achieve better information aggregation for feature interactions.

REFERENCES

[1] Q. Tan, N. Liu, X. Zhao, H. Yang, J. Zhou, and X. Hu, "Learning to hash with graph neural networks for recommender systems," in *International World Wide Web Conference*, 2020, pp. 1988–1998.

[2] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & cross network for ad click predictions," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1–7.

[3] S. C. Geyik, S. Ambler, and K. Kenthapadi, "Fairness-aware ranking in search & recommendation systems with application to linkedin talent search," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2019, pp. 2221–2231.

[4] 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 *Workshop on Deep Learning for Recommender Systems*, 2016, pp. 7–10.

[5] X. Wang, X. He, F. Feng, L. Nie, and T.-S. Chua, "TEM: Tree-enhanced embedding model for explainable recommendation," in *International World Wide Web Conference*, 2018, pp. 1543–1552.

[6] S. Rendle, "Factorization machines," in *IEEE International Conference on Data Mining*, 2010, pp. 995–1000.

[7] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin, "Field-aware factorization machines for CTR prediction," in *ACM Conference on Recommender Systems*, 2016, pp. 43–50.

[8] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 355–364.

[9] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: A factorization-machine based neural network for CTR prediction," in *International Joint Conference on Artificial Intelligence*, 2017, pp. 1725–1731.

[10] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "xDeepFM: Combining explicit and implicit feature interactions for recommender systems," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2018, pp. 1754–1763.

[11] W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, and J. Tang, "AutoInt: Automatic feature interaction learning via self-attentive neural networks," in *ACM International Conference on Information and Knowledge Management*, 2019, pp. 1161–1170.

[12] T. Huang, Z. Zhang, and J. Zhang, "FiBiNET: Combining feature importance and bilinear feature interaction for click-through rate prediction," in *ACM Conference on Recommender Systems*, 2019, pp. 169–177.

[13] Z. Li, Z. Cui, S. Wu, X. Zhang, and L. Wang, "Fi-GNN: Modeling feature interactions via graph neural networks for CTR prediction," in *ACM Internetional Conference on Information and Knowledge Management*, 2019, pp. 539–548.

[14] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," in *International Conference on Learning Representations*, 2016, pp. 1–20.

[15] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017, pp. 1–14.

[16] C. L. P. Chen and Z. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 1, pp. 10–24, 2018.

[17] Y. Qu, B. Fang, W. Zhang, R. Tang, M. Niu, H. Guo, Y. Yu, and X. He, "Product-based neural networks for user response prediction over multi-field categorical data," *ACM Transactions on Information Systems*, vol. 37, no. 1, pp. 1–35, 2018.

[18] R. Wang, R. Shivanna, D. Cheng, S. Jain, D. Lin, L. Hong, and E. Chi, "DCN V2: Improved deep & cross network and practical lessons for web-scale learning to rank systems," in *International World Wide Web Conference*, 2021, pp. 1785–1797.

[19] W. Cheng, Y. Shen, and L. Huang, "Adaptive factorization network: Learning adaptive-order feature interactions," in *AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3609–3616.

[20] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T.-S. Chua, "Attentional factorization machines: Learning the weight of feature interactions via attention networks," in *International Joint Conferences on Artificial Intelligence*, 2017, pp. 3119–3125.

[21] Z. Tao, X. Wang, X. He, X. Huang, and T.-S. Chua, "HoAFM: A high-order attentive factorization machine for CTR prediction," *Information Processing & Management*, vol. 57, no. 6, p. 102076, 2020.

[22] W. Lu, Y. Yu, Y. Chang, Z. Wang, C. Li, and B. Yuan, "A dual input-aware factorization machine for ctr prediction," in *International Joint Conferences on Artificial Intelligence*, 2020, pp. 3139–3145.

[23] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2021.

[24] B. Xiao, E. R. Hancock, and R. C. Wilson, "Graph characteristics from the heat kernel trace," *Pattern Recognition*, vol. 42, no. 11, pp. 2589–2606, 2009.

[25] L. Han, R. C. Wilson, and E. R. Hancock, "Generative graph prototypes from information theory," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 10, pp. 2013–2027, 2015.

[26] L. Bai, L. Cui, L. Rossi, L. Xu, X. Bai, and E. Hancock, "Local-global nested graph kernels using nested complexity traces," *Pattern Recognition Letters*, vol. 134, pp. 87–95, 2020.

[27] H.-N. Tran and E. Cambria, "A survey of graph processing on graphics processing units," *The Journal of Supercomputing*, vol. 74, pp. 2086–2115, 2018.

[28] P. Frasconi, M. Gori, and A. Sperduti, "A general framework for adaptive processing of data structures," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 768–786, 1998.

[29] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.

[30] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *International World Wide Web Conference*, 2015, pp. 1067–1077.

[31] S. Cavallari, E. Cambria, H. Cai, K. C.-C. Chang, and V. W. Zheng, "Embedding both finite and infinite communities on graphs [application notes]," *IEEE Computational Intelligence Magazine*, vol. 14, no. 3, pp. 39–50, 2019.

[32] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[33] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *International Conference on Neural Information Processing Systems*, 2017, pp. 1025–1035.

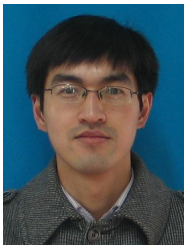
[34] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018, pp. 1–12.

[35] F. Feng, X. He, H. Zhang, and T.-S. Chua, "Cross-GCN: Enhancing graph convolutional network with k-order feature interactions," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–11, 2021.

[36] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Skeleton-based action recognition with directed graph neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7904–7913.

[37] L. Wang, Y. Huang, Y. Hou, S. Zhang, and J. Shan, "Graph attention convolution for point cloud semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 288–10 297.

[38] Z.-M. Chen, X.-S. Wei, P. Wang, and Y. Guo, "Multi-label image recognition with graph convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5172–5181.
[39] D. Teney, L. Liu, and A. Van Den Hengel, "Graph-structured representations for visual question answering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3233–3241.
[40] Y.-H. Pao and Y. Takefuji, "Functional-link net computing: Theory, system architecture, and functionalities," *Computer*, vol. 25, no. 5, pp. 76–79, 1992.
[41] C. L. P. Chen and J. Wan, "A rapid learning and dynamic stepwise updating algorithm for flat neural networks and the application to time-series prediction," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 29, no. 1, pp. 62–72, 1999.
[42] C. L. P. Chen, Z. Liu, and S. Feng, "Universal approximation capability of broad learning system and its structural variations," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 4, pp. 1191–1204, 2019.
[43] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "KGAT: Knowledge graph attention network for recommendation," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2019, pp. 950–958.
[44] X. Yang, L. Prasad, and L. J. Latecki, "Affinity learning with diffusion on tensor product graph," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 28–38, 2013.
[45] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *Computing Research Repository*, vol. abs/1506.00019, pp. 1–38, 2015.
[46] K. Greff, R. K. Srivastava, J. Koutnk, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2017.
[47] I. Chaturvedi, Y.-S. Ong, I. W. Tsang, R. E. Welsch, and E. Cambria, "Learning word dependencies in text by means of a deep recurrent belief network," *Knowledge-Based Systems*, vol. 108, pp. 144–154, 2016.
[48] 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 *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2013, pp. 1222–1230.
[49] 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 *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2016, pp. 255–262.
[50] M. Blondel, A. Fujino, N. Ueda, and M. Ishihata, "Higher-order factorization machines," in *International Conference on Neural Information Processing Systems*, 2016, pp. 3359–3367.
[51] J. Yu, J. Li, Z. Yu, and Q. Huang, "Multimodal transformer with multi-view visual representation for image captioning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 12, pp. 4467–4480, 2020.
[52] J. Yu, C. Zhu, J. Zhang, Q. Huang, and D. Tao, "Spatial pyramid-enhanced netvlad with weighted triplet loss for place recognition," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 2, pp. 661–674, 2020.



Zhifeng Xie received the Ph.D. degree in computer application technology from Shanghai Jiao Tong University, Shanghai, China.

He is currently an Associate Professor with the Department of Film and Television Engineering, Shanghai University, Shanghai, China. He was a Research Assistant with the City University of Hong Kong, Kowloon, Hong Kong. His current research interests include image/video processing, broad learning system, and computer vision.

Wenling Zhang received the B.Eng. degree in computer science and technology from the Nantong University, Nantong, China.

He is currently pursuing the M.Eng. degree with the Department of Film and Television Engineering, Shanghai University, Shanghai, China. His current research interests include recommendation system, broad learning system, and machine learning.

Bin Sheng (Member, IEEE) the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Shatin, Hong Kong.

He is currently an Full Professor with the Shanghai Jiao Tong University, Shanghai, China. He is an Associate Editor of the IEEE Transactions on Circuits and Systems for Video Technology. His current research interests include virtual reality and computer graphics.

Ping Li (Member, IEEE) received the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Shatin, Hong Kong.

He is currently a Research Assistant Professor with The Hong Kong Polytechnic University, Kowloon, Hong Kong. His current research interests include image/video stylization, artistic rendering and synthesis, and creative media. He has one image/video processing national invention patent, and has excellent research project reported worldwide by *ACM TechNews*.

C. L. Philip Chen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1988.

He is currently a Chair Professor and the Dean of the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. Being a Program Evaluator of the Accreditation Board of Engineering and Technology Education (ABET) in the U.S., for computer engineering, electrical engineering, and software engineering programs, he successfully architects the University of Macau's Engineering and Computer Science programs receiving accreditations from Washington/Seoul Accord through Hong Kong Institute of Engineers (HKIE), of which is considered as his utmost contribution in engineering/computer science education for Macau as the former Dean of the Faculty of Science and Technology. He is a Fellow of IEEE, AAAS, IAPR, CAA, and HKIE; a member of Academia Europaea (AE), European Academy of Sciences and Arts (EASA), and International Academy of Systems and Cybernetics Science (IASCYS). He received IEEE Norbert Wiener Award in 2018 for his contribution in systems and cybernetics, and machine learnings. He is also a highly cited researcher by Clarivate Analytics in 2018 and 2019.

His current research interests include systems, cybernetics, and computational intelligence. Dr. Chen was a recipient of the 2016 Outstanding Electrical and Computer Engineers Award from his alma mater, Purdue University (in 1988), after he graduated from the University of Michigan at Ann Arbor, Ann Arbor, MI, USA in 1985. He was the IEEE Systems, Man, and Cybernetics Society President from 2012 to 2013, the Editor-in-Chief of the IEEE Transactions on Systems, Man, and Cybernetics: Systems (2014-2019), and currently, he is the Editor-in-Chief of the IEEE Transactions on Cybernetics, and an Associate Editor of the IEEE Transactions on Fuzzy Systems. He was the Chair of TC 9.1 Economic and Business Systems of International Federation of Automatic Control from 2015 to 2017, and currently is a Vice President of Chinese Association of Automation (CAA).