ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240687

FL-GNN: Efficient Fusion of Fuzzy Neural Network and Graph Neural Network

Boyu Du[a], Jingya Zhou[a,b,c,*], Ling Liu[d] and Xiaolong She[a]

[a]School of Computer Science and Technology, Soochow University
[b]Engineering Lab. of Big Data and Intelligence of Jiangsu Province, Soochow University
[c]State Key Lab. for Novel Software Technology, Nanjing University
[d]School of Computer Science, Georgia Institute of Technology

Abstract. Graph Neural Networks (GNNs) have become an important graph feature learning paradigm that is extensively applied to graph inference tasks. However, GNNs still have limitations in some aspects such as representation capability, interpretability, etc. To this end, this paper introduces a novel hybrid Fuzzy-Logic Graph Neural Network (FL-GNN) that synergistically combines Fuzzy Neural Network (FNN) and GNN to effectively capture and aggregate local information flows within the graph. FL-GNN exhibits three distinctive features. First, we incorporate a specialized structure fuzzy rule to enhance FL-GNN's graph inference capability, surpassing representative GNN models. Second, we augment the interpretability of FL-GNN by integrating analytical exploration methods from two perspectives: Fuzzy Inference System and Message Passing Algorithm (MPA). Lastly, we refine the structure of FL-GNN based on MPA to optimize its calculation complexity and consequently improve learning efficiency. Experimental results show that FL-GNN outperforms existing representative graph neural networks for graph inference tasks.

1 Introduction

Graph is a powerful mathematical tool to model complex relationships between entities or concepts across various domains, such as social networks of people, biological protein structures, and the semantic connections in knowledge graphs. However, graph data often imply hidden, unknown, or missing information regarding edges (connections), vertex/edge attributes, and so forth. The process of inferring such missing information is known as graph inference, which encompasses a variety of tasks, including node classification, link prediction, graph generation, and so on. Graph inference has many applications in various domains, such as traffic-flow prediction, computer vision, and bioinformatics.

Graph Neural Networks (GNNs) [26, 20, 4, 22, 16, 12] are popular algorithms and tools for graph inference. GNNs learn the structure and feature information among the vertices and edges in the graph by iteratively updating their features based on their low-order or high-order neighborhood or even higher-order topology structure. This enables GNNs to extract deep semantics and rich structural information hidden in graph data and apply them to various graph inference tasks.

Fuzzy logic is a type of multi-valued logic that allows for degrees of truth instead of only crisp values. Built on fuzzy logic, a fuzzy inference system [8] consists of a well-designed fuzzy rule base and a fuzzy inference engine and is popularly used to simulate human reasoning and decision-making processes. The fuzzy rule base comprises a set of IF-THEN fuzzy rules, such as "IF temperature is cold, THEN fan speed is slow". The effectiveness of a fuzzy inference system depends on the design of the fuzzy rules, as well as the type and parameters of the membership function, which are typically determined by experts. However, this can be a challenging and tedious task when dealing with complex systems. Several existing efforts have proposed solutions to address this challenge, Buckley and Hayashi [5] propose a fuzzy neural network (FNN), which draws lessons from Artificial Neural Network (ANN) to learn the fuzzy system parameters. FNN has been successfully applied in various fields such as medical image processing [15], time series analysis [18]. reinforcement learning [9], and multimodal sentiment analysis [6]. The state of the art in FNN development can be found in recent review articles [31, 23].

Integrating FNN with GNN has the potential to address three inherent limitations of GNN approaches:

- **Representation capability.** The concept of fuzzification has been extensively studied to improve data representation capabilities. Real-world data often contains uncertainty and fuzziness beyond the scope of traditional crisp values [13, 17]. To address this issue, mechanisms for fuzzification have been developed to capture such fuzziness and uncertainty. To effectively use the fuzzification data, we need to extract features on the fuzzy dimension. However, traditional GNN frameworks cannot do this since they only focus on the original feature dimension and ignore the fuzzy dimension. To overcome this limitation, we develop FL-GNN, which harnesses the advantages of fuzzy features for a more comprehensive data representation solution and utilizes fuzzy rules to extract and highlight fuzzy dimension features.
- **Interpretability.** In conventional GNNs, the correlations between the network's parameter weights and inference process are implicit. In our FL-GNN, the differences in topological structure between communities are explicitly presented in the firing strength distribution. By studying the firing strength distribution, we can identify which rules are valid and which are redundant. This makes interpretability a crucial aspect of FL-GNN as it not only provides a visual perspective for us to observe the local topology information and feature distribution differences of vertices but also provides reliable evidence for us to improve the model structure.

* Corresponding Author. Email: jy_zhou@suda.edu.cn.

• **Flexibility.** Several frameworks have been developed in graph inference, such as Message Passing Algorithm (MPA) [11], which are compatible with most graph inference models. In contrast, fuzzy inference systems offer a more flexible way to establish inference models. Fuzzy rules are not limited to graph-structured data and can be extended to other data structures. For example, Hu et al. [13] directly design fuzzy rules for specialized graph-level task inference and temporal data processing [18].

We conjecture that the proposed FL-GNN will offer some groundwork for developing fuzzy graph neural networks to serve future domain-specific graph applications where more effective fuzzy rules are needed for their graph inference tasks. Designing more sophisticated fuzzy rules enables FL-GNN to handle complex topology structures such as "Hyperedge" and "Simplicial Complex", which will help the model break through the 1-dim WL-test.

One of the big challenges for the efficient fusion of FNN and GNN is the discrepancy between fuzzy rules and graph inference patterns. Fuzzy rules are usually designed to capture human intuition reasoning processes, while graph inference patterns are derived from high-level abstractions of graph structures. As a result, it is challenging to define effective fuzzy rules that work well with graph inference patterns. To this end, we explore a new perspective, where the fuzzy rule is viewed as a paradigm for capturing graph information. By defining a specific fuzzy rule, FNN can handle tasks with diverse data structures, such as the temporal relationship in time series data, the local relationship in image data, or even the topology relationship in graph data. Therefore, we propose FL-GNN, which bridges the gap between FNN and GNN by allowing both fuzzy rules and graph inference patterns to be represented in a unified way.

FL-GNN follows the fuzzy inference process with a human-like mindset and uses MPA to explain it in the GNN field. From the perspective of a fuzzy inference system, FL-GNN utilizes the IF-part of the fuzzy rule to construct the fuzzy relation about the center vertex with its neighborhood vertices, and then the THEN-part outputs the defuzzification inference result as the high-order features for the center vertex. We can directly obtain the structure and semantic information of each vertex in the graph through the firing strength distribution. From the perspective of MPA, the IF-part is designed to capture and aggregate the neighborhood information by combining the t-norm operator and s-norm operator, and the THEN-part is designed to produce the representations by applying a linear transformation. The rule layer of FL-GNN can be treated as a set of aggregation functions to generate abundant aggregation results from different views, i.e., different permutation and combination methods of fuzzy subsets in the rule layer.

The main contributions of this paper are summarized below:

• This paper proposes a novel hybrid model named FL-GNN, which effectively integrates the architecture of FNN and the concept of MPA to handle various graph inference tasks. We provide interpretability of FL-GNN from two perspectives: MPA and fuzzy inference system.

• We further improve FL-GNN to be FL-GNN-A to significantly reduce model complexity while maintaining model performance.

• Extensive experiments verify the inference capability, performance, principle, and interpretability of FL-GNN and FL-GNN-A.

2 Related Work

Some studies have attempted to combine FNN with GNN. Wei et al. [24] pointed out that "similarity" in Few-Shot learning is subjective and uncertain. To improve relationship representations for node

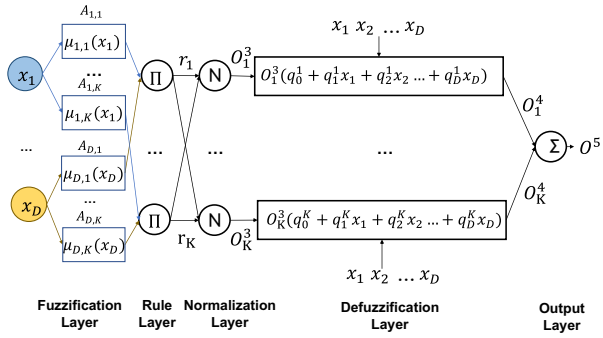

Figure 1. TS-FNN architecture.

classification, they suggest using a membership function instead of SoftMax attention coefficient in the GAT [22] model, although this work introduced fuzzy sets into the GNN model, they did not use fuzzy logic for graph inference. Krleza and Fertalj [17] proposed the FGNN (fuzzy graph neural network) to extract graph information hierarchically from a single graph element to the entire graph structure. FGNN could solve graph-matching problems and exhibited superior robustness to input noise compared to non-fuzzy supervised-learning neural networks.

There are some recent works that focus on modern graph inference tasks. Zhang et al. [30] proposed using fuzzy features for graph contrastive learning and demonstrated the advantages of data fuzzification in representing graph information with abundant experiments. However, this work does not incorporate the core of fuzzy inference systems, including the rule base and inference engine, into such graph inference tasks. Hu et al. [13] presented a fuzzy logic system to deal with graph-level inference tasks. It leverages the graph cluster algorithm to find prototype graphs, and uses features of each prototype graph to generate IF-part of the fuzzy rule. The prototype graph is fed into a GNN-dominated network structure GCPU to generate the network parameters for the THEN-part of the fuzzy rule. However, this model ignores the local topology information and uses graph kernel function and traditional GNN to extract graph information instead of fuzzy logic.

Current works focus on specific types of graph inference tasks and depend on the graph inference paradigm of conventional GNNs, without bringing up a generic graph inference model building on fuzzy logic. FL-GNN as a generic graph inference model, can serve as a tool for extracting graph information and is suitable for various types of graph inference tasks. Meanwhile, the inference principle of FL-GNN is based on fuzzy logic and MPA, enabling it to process fuzzy dimensional features in graph-structured data.

3 Preliminaries

Takagi-Sugeno-FNN (TS-FNN) [19] is one of the most common FNNs, whose architecture is shown in Figure 1, consisting of 5 layers: fuzzification layer, rule layer, normalization layer, defuzzification layer, and output layer.

Given an input vector $\mathbf{x} = [x_1, .., x_D] \in \mathbb{R}^D$, the fuzzy subset corresponding to the ith input variable x_i in the kth rule is denoted as $A_{i,k}$. Let $\mu_{i,k}$ be the corresponding membership function (MF) of $A_{i,k}$. A Gaussian-type MF is defined as

$$\mu_{i,k}(x_i) = e^{\frac{-(x_i - c_{i,k})^2}{2\sigma_{i,k}^2}}, \tag{1}$$

where $c_{i,k}$ is the Gaussian function center, $\sigma_{i,k}$ is the Gaussian function width, and both can be tuned according to the distribu-

tion of input data. When we use singleton fuzzification to fuzzify each input component x_i, the fuzzification result of x_i is denoted by $\mathbf{o_i^1} = [\mu_{i,1}(x_i), ..., \mu_{i,K}(x_i)] \in \mathbb{R}^K$, and the output of the fuzzification layer is given by

$$\mathbf{O^1} = [\mathbf{o_1^1}, \mathbf{o_2^1}, ..., \mathbf{o_D^1}] \in \mathbb{R}^{D\times K}, \tag{2}$$

where K is the number of rules. In TS-FNN, we define the kth rule as follows:

$$\begin{aligned} &IF\ x_1\ is\ A_{1,k}\ AND\ x_2\ is\ A_{2,k}\ ...\ AND\ x_D\ is\ A_{D,k} \\ &THEN\ y^k(\mathbf{x}') = q_0^k + q_1^k x_1 + ... + q_D^k x_D, \end{aligned} \tag{3}$$

where $\mathbf{q}^k = [q_0^k, q_1^k, ..., q_D^k]^\top\ (1 \le k \le K)$ is the trainable parameters of defuzzification layer for the kth rule. "AND" is the t-norm operator in fuzzy logic, denoting the logic conjunction operation between fuzzy subsets. In addition, each t-norm in fuzzy logic has a dual operator named s-norm, which can be written as "OR", denoting disjunction operation between fuzzy subsets. In the following sections, we use $s(\cdot)$ and $t(\cdot)$ as the abbreviation for the s-norm and t-norm functions. As Equation (3) shows, the fuzzy rule is often built by IF-THEN structure. The IF-part of each rule generates a single value in the rule layer named firing strength, reflecting the matching degree of input with the rule, while the THEN-part is responsible for defuzzification, which means converting the fuzzy value into a crisp value. TS-FNN calculates THEN-part on the defuzzification layer.

For Equation (3), if we use "product" as the t-norm operator, then the output of kth rule in the rule layer is $r_k = \prod_{i=1}^{D}\ \mu_{i,k}(x_i),\ (1 \le k \le K)$. We call r_k as the firing strength value. The output of the rule layer is a firing strength vector, i.e.,

$$\mathbf{o^2} = [r_1, r_2, ..., r_K] \in \mathbb{R}^K. \tag{4}$$

The normalization layer is in charge of calculating the weight of the firing strength value of each rule within the whole fuzzy rule base, reflecting its importance of a reasoning process. Then, the kth rule is normalized as follows:

$$O_3^k = \frac{r_k}{\sum_{k'=1}^{K} r_{k'}}. \tag{5}$$

The normalized result of the rule layer is given by

$$\mathbf{o^3} = [O_1^3, O_2^3, \ldots, O_K^3] \in \mathbb{R}^K. \tag{6}$$

The defuzzification layer calculates the THEN-part of the fuzzy rule to output a crisp value directly. The defuzzification result of the kth rule is given by

$$O_k^4 = O_k^3(q_0^k + q_1^k x_1 + ... + q_D^k x_D) = O_k^3(\mathbf{x}'\mathbf{q}^k), \tag{7}$$

where $\mathbf{x}'$ is the input vector in the defuzzification layer. In 1-order TS-FNN, $\mathbf{x}'$ is the ordinary input vector of the fuzzification layer concatenating extra element 1, i.e., $\mathbf{x}' = [1, \mathbf{x}] = [1, x_1, \ldots, x_D] \in \mathbb{R}^{D+1}$. Besides, if $\mathbf{q}^k = [q_0^k]$, the TS-FNN will degenerate to 0-order, and the input vector of the defuzzification layer becomes $\mathbf{x}' = [1] \in \mathbb{R}^1$. Note that the above description just depicts a Multiple Input Multiple Output (MIMO) system. Suppose we adjust the trainable parameter vector to $\mathbf{q}^k \in \mathbb{R}^{(D+1)\times out\_features}$. In that case, the system will become Multiple Input Single Output (MISO), and the $out\_features$ is the dimension of output result. The output vector of the defuzzification layer is denoted by

$$\mathbf{o^4} = [O_1^4, O_2^4, ...O_K^4] \in \mathbb{R}^K. \tag{8}$$

The output layer summarises the total output result of the defuzzification layer, i.e.,

$$O^5 = \sum_{k=1}^{K} O_k^4. \tag{9}$$

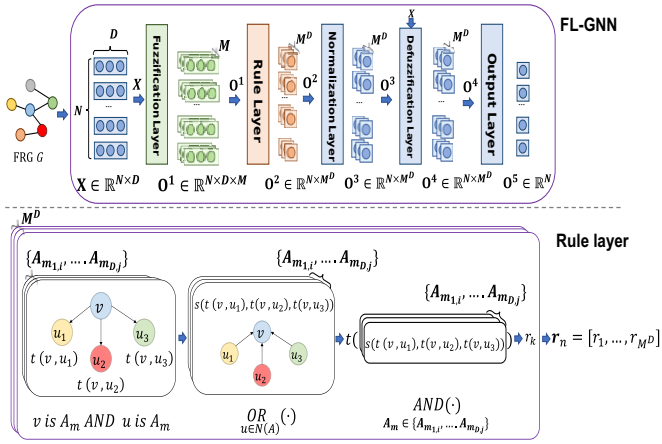

Figure 2. The architecture of FL-GNN.

4 Methodology

We introduce a new concept of fuzzy representation graph (FRG) to describe a graph with fuzzy and uncertain information. Then, we propose FL-GNN to conduct graph inference based on FRG.

4.1 Fuzzy Representation Graph

The information in the real-world graph may be incomplete and fuzzy. Fuzzifying the graph data can capture the fuzziness and prevent information loss. To achieve this, we present FRG to realize the node-level graph fuzzification.

FRG is denoted as $G = (V, E, F_v, F_e, A_v, A_e)$, where V is the set of vertices, E is the set of edges, F_v is the vertex attribute set, F_e is the edge attribute set, A_v is the set of fuzzy subsets for the vertex attribute, and A_e is the set of fuzzy subsets for the edge attribute. For the ith attribute $F_{v,i} \in F_v$, we consider it as a universe of discourse, which has $k_{v,i}$ fuzzy subsets, corresponding to a membership function set $A_{v,i} = \{\mu_n, ..., \mu_{n+k_{v,i}-1}\} \subset A_v$. Similarly, for the jth attribute $F_{e,j} \in F_e$, we also consider it as a universe of discourse, which includes $k_{e,j}$ fuzzy subsets, corresponding to a membership function set $A_{e,i} = \{\mu_m, ..., \mu_{m+k_{e,j}-1}\} \subset A_e$. Let $\phi : V \to 2^{F_v}$, $\psi : E \to 2^{F_e}$ denote the mapping functions for vertex attributes and edge attributes, where 2^{F_v} and 2^{F_e} are the power sets of F_v and F_e, respectively, and then each vertex $v_i \in V$ and edge $e_i \in E$ can be represented by a set of vertex attributes $\phi(v_i) \subset F_v$ and a set of edge attributes $\psi(e_i) \subset F_e$. Meanwhile, we also define two fuzzy subset mapping functions, $\rho : 2^{F_v} \to 2^{A_v}, \sigma : 2^{F_e} \to 2^{A_e}$. Having the above functions, we could fuzzify any vertex and edge by mapping them into a set of fuzzy subsets in FRG, e.g., $\rho(\phi(v_i)) \subset A_v$, $\sigma(\psi(e_i)) \subset A_e$.

4.2 FL-GNN

The architecture of FL-GNN typically follows that of TS-FNN. The difference between MIMO FL-GNN and MISO FL-GNN depends on the trainable parameters $\mathbf{q}^k$. Without loss of generality, we take MISO FL-GNN as an example to illustrate. The architecture of FL-GNN is shown in Figure 2. The upper box displays the main workflow of FL-GNN, while the lower box shows the working procedure of the rule layer for one vertex.

Given an FRG G with N vertices, each vertex's feature vector is denoted by $\mathbf{x}_n = [x_1, \ldots, x_D] \in \mathbb{R}^D\ (1 \le n \le N)$, and each

attribute is assigned with M fuzzy subsets, i.e., $|A_{v,d}| = M$ $(1 \leq d \leq D)$. Let $A_{m_{i,j}}$ denote the jth fuzzy subset of the ith vertex feature $(1 \leq i \leq D, 1 \leq j \leq M)$, and the membership function of $A_{m_{i,j}}$ is denoted by $\mu_{m_{i,j}}$ (we use the abbreviations A_m and μ_m in the remaining part). The input of FL-GNN is the collection of all vertices' feature vectors, $\mathbf{X} \in \mathbb{R}^{N \times D}$. These vectors are first expanded to the fuzzy dimension through the fuzzification layer and we obtain $\mathbf{O}^1 \in \mathbb{R}^{N \times D \times M}$. Then, in FL-GNN we write the kth $(1 \leq k \leq M^D)$ rule of each vertex $v \in V$ as

$$IF \underset{A_m \in (A_{m_{1,a}}, \ldots, A_{m_{D,b}})}{AND} \Big(\underset{u \in N(v)}{OR} (v \; is \; A_m \; AND \; u \; is \; A_m) \Big)$$
$$THEN \; y^k(\mathbf{x}') = (q_0^k + q_1^k x_1 + \ldots + q_D^k x_D), \tag{10}$$

where tuple $(A_{m_{1,a}}, \ldots, A_{m_{D,b}})$ is an element of $S = A_{v,1} \times A_{v,2} \times \cdots \times A_{v,D}$, '$\times$' denotes the Cartesian product, and indexes a, b represent a combination of index results generated by Cartesian product. Hence, set S has a total of M^D tuples, corresponding to M^D rules in the rule layer.

The rule design is based on the concept of MPA to achieve aggregation and update. Specifically, the rule employs a compound logic expression in the IF-part to achieve a *Mixed Aggregation Function* [2], which can aggregate the local information in the fuzzy dimension. In the THEN-part, each vertex updates its self-state based on its firing strength value (aggregation message). The IF-part of Equation (10) corresponds to three steps inside the rule layer described in the lower box of Figure 2: 1) the neighboring vertices of the center vertex v calculate their similarities to v by applying the $t(\cdot)$ operator under the fuzzy subset A_m. Note that the expression $t(v, u_i)$ represents the t-norm operation on the fuzzy values of vertex v and u_i in the fuzzy subset A_m. 2) vertex v uses the $s(\cdot)$ operator to aggregate the similarity messages from its neighboring vertices. 3) the aggregation results from different fuzzy subsets $(A_{m_{1,a}}, \ldots, A_{m_{D,b}})$ are further aggregated by $t(\cdot)$. Then, we obtain the ultimate aggregation outcome corresponding to the firing strength value r_k $(1 \leq k \leq M^D)$. The firing strength vector corresponding to the input vertex vector $\mathbf{x}_n$ is denoted as $\mathbf{r}_n = [r_1, \ldots, r_{M^D}]$ $(1 \leq n \leq N)$ and the output of the rule layer is represented by $\mathbf{O}^2 = [\mathbf{r}_1, \ldots, \mathbf{r}_N] \in \mathbb{R}^{N \times M^D}$. The working procedure of the rule layer is designed to implement an attention mechanism in the fuzzy dimension. Specifically, different combination results in S will generate diverse attention scores to different fuzzy information. After normalization, the normalized firing strength values $\mathbf{O}^3 \in \mathbb{R}^{N \times M^D}$ and all vertices' feature vectors $\mathbf{X}$ are fed into the defuzzification layer to obtain the defuzzification result $\mathbf{O}^4 \in \mathbb{R}^{N \times M^D}$. Finally, the output layer performs the sum operation on $\mathbf{O}^4$ and outputs the final result $\mathbf{O}^5 \in \mathbb{R}^{N \times 1}$.

Here, we formally define the FL-GNN with a stacked structure as follows: First, a single-layer MIMO FL-GNN is defined as $\mathbf{O}^5 = f_\theta(\mathbf{X}, \mathbf{A})$, where $\mathbf{X}$ denotes the input node data, $\mathbf{A}$ denotes the adjacency matrix, θ denotes the trainable parameters in FL-GNN, and $\mathbf{O}^5$ denotes the output of the single layer of FL-GNN. In the following description, we replace $\mathbf{O}^5$ with $\mathbf{H}$ to make the formulas more concise. Then, for the FL-GNN with a stacked structure, we define the output of the lth layer as $\mathbf{H}^l = f_{\theta^l}(\mathbf{H}^{l-1}, \mathbf{A}) + \mathbf{H}^{l-1}$, where we add the residual connection $\mathbf{H}^{l-1}$. Meanwhile, some studies have reported that the FNN structure has the ability to fit nonlinear functions [13]. To enhance the sparsity of the parameters and mitigate overfitting, we incorporate an activation function $\sigma(\cdot)$ of the ReLU family in each layer of FL-GNN. In addition, we also add a BatchNormalization Layer $\mathrm{BN}(\cdot)$ for the stability of the model. Finally, the output of the lth layer is defined as $\mathbf{H}^l = \mathrm{BN}(\sigma(f_{\theta^l}(\mathbf{H}^{l-1}, \mathbf{A}) + \mathbf{H}^{l-1}))$.

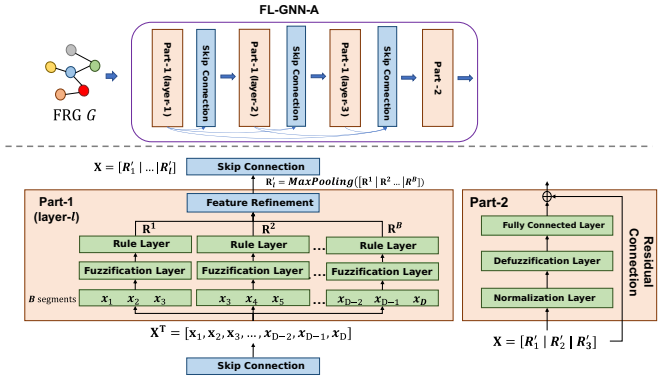

Figure 3. The architecture of FL-GNN-A.

4.3 *Improvement to FL-GNN*

The aforementioned FL-GNN still has limitations. The structural design of the defuzzification and output layers remains complex for graph inference tasks. The additional training parameters and computational steps introduced by the defuzzification and output layers do not bring a proportional performance improvement. To address this, we propose two improvement solutions:

1) We first introduce a sliding window mechanism to achieve dimension reduction. Specifically, we use a sliding window in FL-GNN to split the high-dimensional input into multiple low-dimensional segments, and each individual segment is assigned an independent fuzzification layer and rule layer. This structural design enables multiple FL-GNNs to share the complexity of the monolithic FL-GNN, and reduces the number of rules from M^D to BM^W, where W is the sliding window size ($W \ll D$), and B is the number of segments determined by the sliding window stride size and W. It is important to note that the calculation procedure of firing strength value for each input segment is entirely independent, allowing us to design a parallel architecture that can process all segments simultaneously, thereby improving computation efficiency.

2) To improve the model efficiency for graph inference tasks, we make some changes to FL-GNN's architecture. First, for the stacked structure of FL-GNN, we only stack the fuzzification layer and rule layer, and no longer stack the normalization layer, defuzzification layer, and output layer, which could effectively reduce the computational complexity brought by the defuzzification layer. Second, we add a feature refinement layer that utilizes the MaxPooling function to compress the size of the firing strength vector to reduce information redundancy. In Section 5.3, we conduct a fidelity study to verify that the fuzzy rule with a low value of firing strength is redundant for the inference procession of the model. Third, we add a skip-connection [27] between every two consecutive stacked layers, allowing for merging outputs from previous layers, which can alleviate the problem of excessive information smoothing in stacked structures.

We incorporate two improvement solutions into FL-GNN and elicit FL-GNN-A. For FL-GNN-A, we use a two-part architecture to replace the original 5-layer architecture of FL-GNN to represent the aggregation operation and update operation in turn. The schematic diagram of FL-GNN-A is shown in Figure 3, where the sliding window size is 3, the stride size is 2, and the number of stacked layers is 3. To facilitate the description, we transpose the model input $\mathbf{X} \in \mathbb{R}^{N \times D}$ to $\mathbf{X}^\mathsf{T} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D] \in \mathbb{R}^{D \times N}$, where the

$\mathbf{x}_d \in \mathbb{R}^N$ $(1 \leq d \leq D)$ denotes the vector consisting of the dth feature of all vertices.

In the part-1 of FL-GNN-A, we first utilize the sliding window to split the vertex feature along the feature dimension. These segments are then independently processed in the fuzzification layer and rule layer to generate the firing strength segment $\mathbf{R}^b = [\mathbf{r}_1^b, \ldots, \mathbf{r}_N^b] \in \mathbb{R}^{N \times M^W}, (1 \leq b \leq B)$, where $\mathbf{r}_n^b \in \mathbb{R}^{M^W}, (1 \leq n \leq N)$ denotes the firing strength vector for the nth vertex in the bth segment. The output of part-1 is denoted by $\mathbf{R}_l' = [\mathbf{R}^1, \ldots, \mathbf{R}^B] \in \mathbb{R}^{N \times BM^W}, (1 < l < L)$, where L is the number of stacked layers. For the sake of the narrative, here we ignore the feature refinement layer. $\mathbf{R}_l'$ is then fed into the skip connection layer, where the outputs of all previous layers will be concatenated as $\mathbf{X} = [\mathbf{R}_1', \ldots, \mathbf{R}_l']$ and sent to the next part-1 layer. In part-2, the output of the last skip connection layer passes through the normalization layer, defuzzification layer, and fully connected layer sequentially to generate the final outcome. Although FL-GNN-A does not follow the conventional structure of FNN, it still maintains the fundamental structure of the fuzzification layer and the rule layer. This implies that FL-GNN-A can effectively mitigate noise and extract fuzzy information from a graph. It is a trade-off between the model's computational efficiency and inference capability.

4.4 FL-GNN and MPA

Let us shift our focus to the Message Passing Neural Network (MPNN), which is the generalized paradigm for most GNNs. MPA is a crucial phase of MPNN. The major steps of MPA can be divided into two steps: the message aggregation and the message update. In the message aggregation step, each vertex combines the messages from its neighbors using aggregation mechanisms such as sum, $concatenate$, $attention$, etc. Then, in the message update step, each vertex updates its own state using an update function.

On the whole, FL-GNN generates M^D different aggregation functions (fuzzy rules) in the IF-part, and the THEN-part in each fuzzy rule acts as an update function.

The calculation process of the IF-part in each fuzzy rule as the *Mixed Aggregation Function* denoted by $f : [0,1]^n \to [0,1]$. This function takes real arguments from the closed interval $\mathbb{I} = [0,1]$ and produces a real value in the same interval. The components of f, t-norm and s-norm operators, are the conjunctive aggregation function and disjunctive aggregation function, respectively [2]. No matter whether t-norm or s-norm, they both possess $symmetric$, namely, $permuation\ \ equivariant$, satisfying the basic inductive bias of GNN. As per the conclusion of Corso et al. [7]: "*In order to discriminate between multisets of size n whose underlying set is* $\mathbb{R}$*, at least n aggregators are needed.*" FL-GNN obtains rich aggregation functions by combining different fuzzy subsets. Therefore, we believe that the rule layer provides sufficient inference capability for FL-GNN.

The principle of update function of THEN-part is similar to the attention mechanism, where the original feature of the vertex first undergo a linear transform and then multiplied by the attention score (firing strength value) to obtain the new vertex state.

4.5 Complexity Analysis

Here we conduct the complexity analysis with FL-GNN and FL-GNN-A. The signal layer of FL-GNN comprises five calculation steps.

Step 1: The Fuzzification Layer will expand the vertices feature $\mathbf{X} \in \mathbb{R}^{N \times D}$ to the fuzzy dimension and output $\mathbf{O}^1 \in \mathbb{R}^{N \times D \times M}$. the computational complexity of this operation is $\mathcal{O}(NMD)$.

Step 2: Then, in the Rule Layer for each fuzzy rule, the computational complexity of innermost operation $(v\ is\ A_m\ AND\ u\ is\ A_m)$ is $\mathcal{O}(1)$, the inner operation $OR_{u \in N(v)}(\cdot)$ will aggregate the fuzzy feature of all vertices, its computational complexity is $\mathcal{O}(|E|MD)$. The computational complexity of outer operation $AND_{A_m \in (A_{m_{1,a}}, \ldots, A_{m_{D,b}})}(\cdot)$ is $\mathcal{O}(D)$. In the Rule Layer, each vertex will generate M^D fuzzy rules, and the complexity of calculating the firing strength of all fuzzy rules is $\mathcal{O}(NDM^D)$. However, we have noticed that some elements in the result of Cartesian product $S = A_{v,1} \times A_{v,2} \times \cdots \times A_{v,D}$ have the identical prefix. Therefore, we can store the calculation results of the common prefix via iterative fuzzy rule computation to avoid duplicate calculations, then the computation complexity could be reduced to $O(\frac{NM^2(1-M^{D-1})}{1-M})$. Finally, the computational complexity in the Rule Layer is $\mathcal{O}(|E|MD + O(\frac{NM^2(1-M^{D-1})}{1-M}))$.

Step 3: The Normalization Layer will carry on normalization operation for all M^D firing strength of the fuzzy rule, the complexity is $\mathcal{O}(M^D)$.

Step 4: In Defuzzification Layer, the complexity of $\mathbf{x}'\mathbf{q}^k$ is $\mathcal{O}(DD')$, where we use D' denotes the output dimension. Thus, for M^D fuzzy rules the complexity is $\mathcal{O}(M^D DD')$, and the total complexity is $\mathcal{O}(NM^D DD')$.

Step 5: In the Output Layer, we accumulate the fire strength of all fuzzy rules, with a computational complexity of $\mathcal{O}(NM^D)$.

Finally, we obtain the computational complexity of FL-GNN: $\mathcal{O}(MD(N+|E|))+\mathcal{O}(M^D(N(DD'+1)+1)+\frac{NM^2(1-M^{D-1})}{1-M})$.

For the FL-GNN-A, the number of fuzzy rules will be reduced to BM^W through sliding windows, so we can reduce the computational complexity of the Rule Layer to be $\mathcal{O}(|E|MD + \frac{NBM^2(1-M^{W-1})}{1-M}))$. Meanwhile, when using FL-GNN-A in practice, we usually directly use a fully connected layer instead of the Normalization Layer, Defuzzification Layer, and Output Layer, because the firing strength vector is sufficient to reflect the local structure information, so we can directly use it as the representation information of the vertex in the graph. Thus, the computational complexity of the last three layers is reduced to $\mathcal{O}(NBM^W D')$. Finally, the computational complexity of FL-GNN-A is $\mathcal{O}(MD(N + |E|)) + \mathcal{O}(NB(M^W D' + \frac{M^2(1-M^{W-1})}{1-M}))$.

5 Experiments

5.1 Datasets

We conduct experiments on multiple datasets to evaluate the performance of the model for various graph inference tasks. For the node-level task, we select three small node classification datasets, including Cora, Citeseer, and a Pubmed [28], all of them use F1-micro as the evaluation metric. A medium dataset ogbn-protines [14], uses AUC-ROC as its evaluation metric. A large-scale node classification dataset Reddit [12], uses F1-micro as its evaluation metric. For the graph-level task, we select two small-scale graph-level node classification/regression datasets, including ogbg-molsol (regression), and ogbg-molfreesolv (regression). A medium-scale dataset ogbg-molhiv (classification), uses AUC-ROC as its evaluation metric. A large-scale dataset ogbg-molpcba (classification) [14], uses AP (average precision) as its evaluation metric. All the performance results are presented in the form of mean value $\pm$ standard deviance.

Table 1. Performance on node-level datasets.

	Cora (F1-micro)	CiteSeer (F1-micro)	Pubmed (F1-micro)	ogbn-proteins (AUC-ROC)	Reddit (F1-micro)
FL-GNN-A	0.8252±0.0037	**0.7254±0.0063**	**0.7860±0.0013**	**0.7897 ±0.0012**	0.9521±0.0015
GraphSage-mean	0.8144±0.0091	0.6994±0.0083	0.7660±0.0059	0.7768 ±0.0020	0.9501±0.0023
GAT	0.8202±0.0056	0.7188±0.0067	0.7775±0.0066	0.7616 ± 0.0039	0.9429±0.0074
GCN	0.8220±0.0008	0.7034±0.0055	0.7762±0.0019	0.7251±0.0035	0.9311±0.0131
TransformerConv	0.8002±0.0039	0.6712±0.0063	0.7732±0.0012	0.7873 ± 0.0054	**0.9550±0.0067**
GATv2	0.8174±0.0036	0.7208±0.0093	0.7724±0.0055	0.7608 ± 0.0015	0.9220±0.0445
GIN	0.7408±0.0205	0.6260±0.0128	0.7376±0.0128	0.7569 ± 0.0054	0.9218±0.0173
GTCN	**0.8297±0.0013**	0.7201±0.0898	0.7831±0.0048	0.7673 ± 0.0014	0.9373±0.0014
GTAN	0.8038±0.0110	0.7127±0.0046	0.7246±0.0076	0.7601 ± 0.0135	0.9401±0.0135

Table 2. Performance on graph-level datasets.

	ogbg-molfreesolv (RMSE)	ogbg-molesol (RMSE)	ogbg-molhiv (AUC-ROC)	ogbg-molpcba (AP)
FL-GNN-A	**1.7075±0.0052**	**0.8135±0.0416**	**0.7824±0.0157**	0.2468±0.0032
GraphSage-mean	2.0390±0.0631	0.8437±0.0765	0.7568±0.2610	0.2333±0.0045
GAT	2.0978±0.1933	0.8413±0.0251	0.7754±0.0164	0.2370±0.0632
GCN	1.8471±0.0808	0.8188±0.0249	0.7673±0.2187	0.2287±0.0113
TransformerConv	1.9935±0.1166	0.8671±0.0672	0.7652±0.2231	0.2433±0.0124
GATv2	1.9905±0.1052	0.8313±0.0255	0.7571±0.0083	0.2455±0.0113
GIN	2.0219±0.0367	0.8364±0.0173	0.7798±0.0343	0.2423±0.0299
E-SPN	1.7903±0.0471	0.8693±0.0569	0.7710±0.0120	0.2348±0.0087
GSN	1.8096±0.0033	0.8239±0.0131	0.7703±0.0174	**0.2508±0.0230**

5.2 *Performance Comparison*

Experiment Settings. For FL-GNN-A, we choose the "product" t-norm operator and replace its dual operator (s-norm operator) with the "mean" average operator by considering that the calculation of s-norm cannot be performed in parallel. We choose several widely-used GNNs as the baseline models, including Graph Isomorphism Network (GIN) [26], TransformerConv [20], GATv2 [4], GAT [22], Graph Convolutional Network (GCN) [16], and GraphSAGE [12]. For the models with the multi-heads mechanism, such as TransformerConv and GAT, we set 4 heads for them. All the GNN models mentioned above are implemented through the model interface provided in Fey and Lenssen [10]. Moreover, we also select some models specifically designed for a certain type of graph inference task, including GSN[3] and E-SPN[1], which are used for graph-level inference task. For the node-level inference task, we added two variants of GTNet[25], namely GTCN and GTAN. On the dataset Reddit, all models are trained in batches using the NeighborSampling scheme [12], and the sample sizes are set to $layer_1 = 35$ and $layer_2 = 20$ for both the 1st-order and 2nd-order neighbors, and on the dataset ogbn-proteins, the sample sizes of the 1st-4th order neighbors are set to $layer_1 = 40$, $layer_2 = 35$, $layer_3 = 15$, $layer_4 = 5$. For the graph-level datasets, we select "sum" as the readout function.

Experimental Results. The results for node-level and graph-level graph inference tasks are reported in Tables 1 and 2, respectively. We observe that FL-GNN-A substantially possesses sufficient graph inference capability compared to popular GNN models. FL-GNN-A can handle both node-level and graph-level inference tasks regardless of their scales, indicating that the model structure of FL-GNN-A is suitable for common graph inference tasks. FL-GNN-A achieves great performance on most graph-level datasets except GSN on the ogbg-molpcba dataset. In this case, FL-GNN-A's performance is second only to GSN. Meanwhile, FL-GNN-A also achieves superior performance for the node-level tasks and exceeds almost all models except TransformerConv on the Reddit dataset and GTCN on the Cora dataset, where FL-GNN-A performs as the second-best with a minimal gap compared to TransformerConv and GTAN. The results show that few existing GNNs can always maintain good performance for one task on multiple datasets, let alone for both graph-level and node-level inference tasks. For example, TransformerConv performs the second-worst for the node-level task on the datasets Cora and CiteSeer. GCN achieves the second-best performance for the graph-level task on small-scale datasets, but has the lowest performance on the large-scale dataset. In addition, GNN models that focus on specific types may not always achieve good model performance. For example, GCTN performs poorly on large-scale node-level inference tasks, while GSN performs poorly on small-scale graph-level inference tasks. We interpret that the outstanding performance of FL-GNN-A is due to the finer-grained features provided by fuzzification. This not only enhances the representation ability of the original features but also offers a certain level of noise resistance.

5.3 *Fidelity Study*

The interpretability of FNN in the inference process is a significant advantage over traditional deep learning models. The interpretability of FL-GNN is reflected in the firing strength distribution that is explicitly related to the graph inference process of FL-GNN. This means that fuzzy rules with low firing strength values are redundant for the FL-GNN inference process. To verify this viewpoint, we conduct a fidelity experiment and introduce GNNExplainer [29] as a control group. GnnExplainer is a model-free method that can provide a reliable explanation for GNN that conforms to the message-passing paradigm. GnnExplainer takes a trained GNN model and its prediction results as input and outputs the subgraphs and feature sets that truly affect the prediction results.

Experiment Settings. We set up two groups of experiments. In the first group, we pre-train the FL-GNN-A on the Pumbed, Cora, and CiteSeer datasets. Then, in the inference phase of the pre-trained model, for each node class, we separately compute the average of its firing strength values for the fuzzy rules in each stacked layer and mark those with lower values according to a ratio (e.g., Pubmed-10% indicates that we mask the fuzzy rule whose firing strength value is in the bottom 10%). We then re-train the model while masking the marked fuzzy rules. Finally, we calculate the model score using test data as its fidelity. In the second group, we feed the FL-GNN-A

Table 3. The fidelity experimental results.

GNNExplainer-Pubmed	Pubmed-10%	Pubmed-15%	Pubmed-20%	Pubmed-30%	Pubmed-40%
0.9397 ± 0.0004	0.7833 ± 0.0013	0.7972 ± 0.0073	0.8272 ± 0.0161	0.9660 ± 0.0018	0.9962 ± 0.0057
GNNExplainer-Cora	Cora-30%	Cora-40%	Cora-60%	Cora-70%	Cora-80%
0.9608 ± 0.0025	0.8058 ± 0.0097	0.8299 ± 0.0052	0.8756 ± 0.0010	0.98639 ± 0.0060	0.9862 ± 0.0039
GNNExplainer-CiteSeer	CiteSeer-5%	CiteSeer-7%	CiteSeer-9%	CiteSeer-10%	CiteSeer-15%
0.9410 ± 0.0013	0.7203 ± 0.0038	0.7329 ± 0.0015	0.8003 ± 0.0026	0.9370 ± 0.0058	0.9989 ± 0.0064

model pre-trained on the Pubmed, Cora, and CiteSeer datasets into the GNNExplainer to extract the subgraphs as well as the feature sets that are more useful for the inference task. We then use the extracted information for inference on the test dataset to calculate the model score as its fidelity of GNNExplainer. For GNNExplainer, the regularization hyperparameters for subgraph size, laplacian, and feature explanation in GNNExplainer are 0.005, 0.5, and 0.1, respectively.

Experimental Results. The experimental results are presented in Table 3. We observe that increasing the mask ratio of the fuzzy rule has a positive impact on the model's performance, indicating a strong correlation between the firing strength distribution and the graphical inference process of FL-GNN-A. Thus, the masking of fuzzy rules with low firing strength values does not negatively impact the inference process of our model. We argue that masking invalid fuzzy rules can be used as a training trick for FL-GNN. Although this method may introduce labeling information, similar to the multi-stage training methods [21] that add high-confidence training set data and validation set data to the next training set, FL-GNNs also provide guidance on the next inference based on its own inference results. Therefore, we believe that exploring the interpretability of FL-GNN will help to unlock its full potential. Besides, we can see that the information extracted by GNNExplainer improves the model significantly.

5.4 Ablation Study

Table 4. The performance comparison in ablation study.

model	feature refine	hidden	ogbg-molhiv
FL-GNN	None	5	0.7091±0.0171
		7	0.7121±0.0165
		9	0.7607±0.0165
FL-GNN-A	None	32	0.7402±0.0269
	MaxPooling-1D-70%		0.7555±0.0023
	MaxPooling-1D-30%		0.7487±0.0043
	None	64	0.7604±0.0080
	MaxPooling-1D-70%		0.7648±0.0107
	MaxPooling-1D-30%		0.7596±0.0109
	None	128	0.7781±0.0043
	MaxPooling-1D-70%		0.7636±0.0185
	MaxPooling-1D-30%		0.7577±0.0251
FL-GNN-*	None	32	0.6442±0.0106
		64	0.6244±0.0201
		128	0.6443±0.0198
model	feature refine	hidden	Cora
FL-GNN	None	5	0.3871±0.1181
		7	0.5204±0.1048
		9	0.6102±0.0581
FL-GNN-A	None	32	0.8230±0.0016
	MaxPooling-1D-70%		0.8200±0.0048
	MaxPooling-1D-30%		0.8238±0.0044
	None	64	0.8240±0.0023
	MaxPooling-1D-70%		0.8252±0.0037
	MaxPooling-1D-30%		0.8242±0.0061
	None	128	0.8202±0.0021
	MaxPooling-1D-70%		0.8224±0.0049
	MaxPooling-1D-30%		0.8130±0.0050
FL-GNN-*	None	32	0.3691±0.0349
		64	0.4864±0.0182
		128	0.4437±0.0261

To verify whether FL-GNN and FL-GNN-A have a positive impact on graph inference, we conduct the ablation experiment.

Specifically, we compare performance among FL-GNN, FL-GNN-A, and FL-GNN-* on the ogbg-molhiv and Cora datasets, where FL-GNN-* is a variant of FL-GNN-A that removes the fuzzy inference module and retains only the necessary fully connected layer. We use the feature-refinement module to extract 70% and 30% of the firing strength values, i.e., 70% and 30% of the length of the original firing strength vector will be retained. For FL-GNN, when we set the number of membership functions for each feature greater than or equal to 3 and the number of hidden dimensions exceeds 10, the dimension explosion will occur. Thus, in this experiment, we set the number of hidden units for FL-GNN to be much smaller than FL-GNN-A and FL-GNN-*. As Table 4 shows, compared to FL-GNN-*, FL-GNN achieves a significant performance improvement on the Cora and ogbg-molhiv datasets. This improvement is attributed to the fuzzy inference structure, which provides FL-GNN with powerful graph inference ability. Compared to FL-GNN, FL-GNN-A achieves further improvement. We conclude that FL-GNN-A strikes a significant balance between model performance and computational overhead. For FL-GNN, by contrast, the improvement in model performance is disproportionate to the increase in computational overhead of the model. In addition, when observing the effect on the MaxPooling1D function, we find that the MaxPooling1D function has only a minor impact on the final model performance. However, it helps to accelerate the inference and training process by $1.2 \sim 1.3$ times while also significantly reducing the trainable parameters of the model.

6 Conclusion

In this paper, we investigated the efficient fusion of FNN and GNN and proposed a novel hybrid model, named FL-GNN, to use fuzzy logic for graph inference tasks, which provides a novel insight into graph inference using fuzzy logic. Furthermore, we improved FL-GNN from two perspectives of dimension reduction and model architecture and obtained FL-GNN-A. The experimental results showed that our proposed model can achieve better inference performance on multi-scale datasets and provide good interpretability at the same time. In the future, we intend to introduce fuzzy integrals to deal with more complex and multi-modal graph data.

Acknowledgements

This work is supported by the Natural Science Foundation of China under grant 61972272, the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under grant 21KJA520008, Qinlan Project of Jiangsu Province of China, Postgraduate Research & Practice Innovation Program of Jiangsu Province under grant SJCX23_1660, Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions. Ling Liu is partially supported by USA NSF CISE grants 2302720, 2312758, 2038029, an IBM faculty award, and a grant from CISCO Edge AI program.

References

[1] R. Abboud, R. Dimitrov, and İ. İ. Ceylan. Shortest path networks for graph property prediction. In B. Rieck and R. Pascanu, editors, *Learning on Graphs Conference, LoG 2022, 9-12 December 2022, Virtual Event*, volume 198 of *Proceedings of Machine Learning Research*, page 5. PMLR, 2022.

[2] G. Beliakov, S. James, and J. Wu. *Discrete Fuzzy Measures - Computational Aspects*, volume 382 of *Studies in Fuzziness and Soft Computing*. Springer, 2020. ISBN 978-3-030-15304-5.

[3] G. Bouritsas, F. Frasca, S. Zafeiriou, and M. M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(1):657–668, 2023.

[4] S. Brody, U. Alon, and E. Yahav. How attentive are graph attention networks? In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[5] J. J. Buckley and Y. Hayashi. Fuzzy neural networks: A survey. *Fuzzy Sets and Systems*, 66(1):1–13, 1994. ISSN 0165-0114.

[6] I. Chaturvedi, R. Satapathy, S. Cavallari, and E. Cambria. Fuzzy commonsense reasoning for multimodal sentiment analysis. *Pattern Recognit. Lett.*, 125:264–270, 2019.

[7] G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Velickovic. Principal neighbourhood aggregation for graph nets. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[8] E. Czogala and J. M. Leski. *Fuzzy and Neuro-Fuzzy Intelligent Systems*, volume 47 of *Studies in Fuzziness and Soft Computing*. Springer, 2000. ISBN 978-3-662-00389-3.

[9] B. Fang, C. Zheng, H. Wang, and T. Yu. Two-stream fused fuzzy deep neural network for multiagent learning. *IEEE Transactions on Fuzzy Systems.*, 31(2):511–520, 2023.

[10] M. Fey and J. E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[11] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017.

[12] W. L. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1024–1034, 2017.

[13] F. Hu, Z. Deng, G. Wang, Z. Xie, K.-S. Choi, and S. Wang. Graph fuzzy system for the whole graph prediction: Concepts, models and algorithms. *IEEE Transactions on Fuzzy Systems*, pages 1–15, 2023.

[14] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[15] M. Kaur and D. Singh. Fusion of medical images using deep belief networks. *Clust. Comput.*, 23(2):1439–1453, 2020.

[16] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[17] D. Krleza and K. Fertalj. Graph matching using hierarchical fuzzy graph neural networks. *IEEE Transactions on Fuzzy Systems.*, 25(4): 892–904, 2017.

[18] C. Luo, C. Tan, X. Wang, and Y. Zheng. An evolving recurrent interval type-2 intuitionistic fuzzy neural network for online learning and time series prediction. *Appl. Soft Comput.*, 78:150–163, 2019.

[19] S. D. Rajurkar and N. K. Verma. Developing deep fuzzy network with takagi sugeno fuzzy inference system. In *2017 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2017, Naples, Italy, July 9-12, 2017*, pages 1–6. IEEE, 2017.

[20] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In Z. Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 1548–1554. ijcai.org, 2021.

[21] K. Sun, Z. Lin, and Z. Zhu. Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 5892–5899. AAAI Press, 2020.

[22] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[23] L. Wang. Fast training algorithms for deep convolutional fuzzy systems with application to stock index prediction. *IEEE Transactions on Fuzzy Systems*, 28(7):1301–1314, 2020.

[24] T. Wei, J. Hou, and R. Feng. Fuzzy graph neural network for few-shot learning. In *2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020*, pages 1–8. IEEE, 2020.

[25] N. Wu and C. Wang. Gtnet: A tree-based deep graph learning architecture. *CoRR*, abs/2204.12802, 2022.

[26] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[27] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka. Representation learning on graphs with jumping knowledge networks. In J. G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5449–5458. PMLR, 2018.

[28] Z. Yang, W. W. Cohen, and R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In M. Balcan and K. Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 40–48. JMLR.org, 2016.

[29] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 9240–9251, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/d80b7040b773199015de6d3b4293c8ff-Abstract.html.

[30] C. Zhang, Y. Lin, C. L. P. Chen, H. Yao, H. Cai, and W. Fang. Fuzzy representation learning on graph. *IEEE Trans. Fuzzy Syst.*, 31(10):3358–3370, 2023.

[31] Y. Zheng, Z. Xu, and X. Wang. The fusion of deep learning and fuzzy systems: A state-of-the-art survey. *IEEE Transactions on Fuzzy Systems.*, 30(8):2783–2799, 2022.