Iranian Journal of Fuzzy Systems

Volume 22, Number 2, (2025), pp. 129-146

Original paper

University of Sistan and Baluchestan

Enhanced high-dimensional data classification by combining fuzzy learning integration and graph transformers

S. Bastami,[1], R. P. Mohammadiani,[2], M. B. Dowlatshahi,[3] and A. Abdollahpouri,[4]

[1,2,4] *Department of Computer Engineering, University of Kurdistan, Sanandaj, Iran*
[3] *Department of Computer Engineering, University of Lorestan, Khorramabad, Iran*

sajad.bastami@uok.ac.ir, r.pirmohamadiani@uok.ac.ir, dowlatshahi.mb@lu.ac.ir, Abdollahpouri@uok.ac.ir

Abstract

Graph neural networks and fuzzy models offer effective and practical methods for solving various tasks at the large-scale graph level. Large-scale graph embedding based on deep methods and fuzzy models is categorized into fusion and integration. Feature extraction and graph structure at the local and global levels are based on augmented graph fusion. In fusion-based graph embedding, the fuzzy model is used as an activation function based on an aggregated process. In some cases, the fusion of graph neural network methods with fuzzy systems has been successful. However, no effective methods have been developed for integrating fuzzy models with deep methods. Two main issues are associated with this integration: (1) computational complexity due to the exponential increase in fuzzy rules with the number of features, and (2) the complexity of the solution space due to the combination of fuzzy regression rules between inputs and outputs. Additionally, modeling at the large-scale graph level using linear regression and graph neural networks is not sufficient. Therefore, this paper proposes a feature and structure combination method at the local and global levels using a combination of fuzzy modeling and graph transformers, an integrated deep learning technique called Fuzzy Graph Transformer (FuzzyGT). We conducted experiments on deep learning graph datasets to compare with the proposed model. Our method achieved the best results compared to other advanced models.

Keywords: Graph transformers, fuzzy systems, deep learning, social network analysis.

1 Introduction

Deep learning is a suitable method for addressing downstream tasks in large-scale graphs. These models learn a substantial number of features and node structures [3], discovering appropriate representations [4] in hidden space and distributing information at the local level of the graph [35]. Nevertheless, most real-world raw data connections are defined locally between distinct levels of the graph. Therefore, analyzing data at the global graph level is useful for utilizing all possible information. Graph Neural Networks (GNNs) are deep learning models designed to solve end-to-end downstream graph tasks and explicitly extract high-level representations. Hence, GNNs address the graph embedding problem through an autoencoder framework. These methods are presented to tackle machine learning issues in graph embedding modeling [46].

Despite significant advancements in graph neural networks, these networks are notably susceptible to over-smoothing and are sensitive to training data. Noisy or incomplete data can adversely impact the performance of graph neural networks, leading to unreliable models [29]. Generally, fuzzy systems combine data or features and enhance prediction based on this combination [26]. To enhance the robustness of deep graph-based models against noisy and incomplete data, deep methods are combined with fuzzy systems [12] at the node or feature level. Hybrid models extract the structure and features of nodes at both local and global levels of the graph [10]. By leveraging the computational capacity of deep graph techniques and the stability of fuzzy methods, integrating these approaches offers a potential enhancement in model responsiveness when addressing issues with noisy and incomplete data [14].

Corresponding Author: R. P. Mohammadiani
Received: January 2025; Revised: March 2025; Accepted: April 2025.
https://doi.org/10.22111/ijfs.2025.50945.8999

This improvement in graph neural network systems can be attributed in part to the adaptability of fuzzy systems [25] to deep learning architectures. Consequently, GNNs can leverage the approximation capabilities of convolutional networks, along with the interpretability offered by fuzzy (if-then) rules. The combination of adaptive fuzzy systems and graph neural networks has attracted researchers' attention. However, the hybrid approach based on GNN models and fuzzy logic systems [16] has not been implemented at the graph-wide level; in other words, the applications of efficient graph transformer models with fuzzy logic systems have not been fully explored in the domain of big data. In some cases, the combination of the two models mentioned above is not implemented. However, the methods that do combine these models focus solely on transforming membership value representations using a fuzzy system, which is considered a form of set-based inference. Examples of such methods include: Dalibor et al. [20] presented a model examining pattern recognition in graphs, providing the ability to execute noise-resistant fuzzy graph queries; and Tham et al. [44] introduced a noise-reduced attention-based graph autoencoder mechanism integrated with a noise-reducing fuzzy neural network.

Most research in the domain of combining graph neural networks (GNNs) and fuzzy systems follows a local graph-level combination pattern. In this pattern, adaptive methods and fuzzy inference [33] are used as activation functions, and data are extracted from various neighborhood layers [30]. However, the results of these techniques are not integrated into the global graph-level learning process. In the proposed model, we directly combine the fuzzy system with graph transformers, unlike previous methods, which only utilized membership functions and node neighborhood information [9], as shown in (See Figure 1) As mentioned, two main challenges exist in integrated combination methods: (1) the increased computational cost of fuzzy rules, and (2) the challenge of generalizing fuzzy subsets to fit complex models [1] between inputs and outputs. So far, no method based on global graph-level exploration has succeeded in seamlessly combining graph transformers [53] and fuzzy systems for deep learning applications in large data sets. In this paper, we propose a new unified model that leverages fuzzy learning to enhance graph representation learning. This approach allows the graph transformer to effectively address and overcome the two challenges. The fuzzy learning technique in the graph transformer improves the expressiveness of node representation at the global graph level. The key aspects of our contribution are presented below:

- We propose a novel framework that seamlessly combines fuzzy learning and graph transformers. Fuzzy rules are incorporated into the node representation mechanism at both local and global levels, enabling the model to effectively capture both local structures and global patterns within graph data.
- To address the challenge of scaling fuzzy rules to large datasets, we introduce efficient sampling methods that significantly reduce the model's parameters while maintaining its performance.
- We enhance the model's predictive performance by incorporating a fuzzy consequence layer within each domain defined by the fuzzy system. This layer generalizes a polynomial function, enabling a more accurate representation of the correlation between the input and output variables.
- We integrate fuzzy information directly into the graph transformer through a fuzzy input layer. This allows the model to better adapt to real-world data and reduce the discrepancy between predictions and observations.
- We evaluate the effectiveness of the proposed Fuzzy Graph Transformer (FuzzyGT) through a series of comprehensive experiments on five widely recognized machine learning benchmark datasets: Cora, Pubmed, Citeseer, ogbn-arxiv, and ogbn-products. The results consistently demonstrate that FuzzyGT outperforms current state-of-the-art methods, establishing it as a leading approach in the field.

The organization of this paper is as follows: Section 2 presents an overview of the foundational concepts and related works. Section 3 presents the proposed model. The experimental validation and analysis of the results are discussed in Section 4. Finally, Section 5 concludes the paper.

2 Background and related works

In this section, we introduce the mechanism of the Graph Transformer (GT) and review recent innovations in combining fuzzy learning with graph transformers. We will also discuss the potential applications of this combination and the challenges ahead.

Deep learning, through multi-layer computational models, facilitates automatic [11] feature extraction from raw data at various levels of abstraction. This capability has made deep learning a cornerstone of complex tasks such as image classification, natural language processing, and pattern recognition. However, designing optimal deep neural network (DNN) architectures remains a significant challenge due to the complexity of manually tuning network parameters. The process often involves extensive experimentation and expert knowledge, making it inefficient and resource-intensive.

To address these challenges, Neural Architecture Search (NAS) has emerged as a promising solution by automating the discovery of new architectures, particularly for Convolutional Neural Networks (CNNs). NAS aims to optimize the structure of neural networks without requiring manual intervention, thereby reducing the dependency on human expertise. Despite its advantages, NAS faces substantial computational complexity and cost, limiting its applicability in real-world scenarios where computational resources are constrained [34].

Recent advancements in NAS have focused on enhancing efficiency and effectiveness through state-of-the-art (SOTA) methodologies. Some notable approaches include the following. Neural Architecture Search with Progressive Evaluation and Sub-Population Preservation [49] incorporates progressive evaluation techniques to improve stability in architecture selection. By preserving diversity among candidate architectures, this approach enhances robustness and prevents premature convergence to suboptimal solutions. A Gradient-Guided Evolutionary Neural Architecture Search [50] integrates evolutionary search mechanisms with gradient-based updates, allowing for a more guided and efficient exploration of the search space. The combination of gradient information and evolutionary strategies accelerates convergence while maintaining flexibility in architecture optimization. Score Predictor-Assisted Evolutionary Neural Architecture Search introduces a score predictor that estimates the performance of candidate architectures, reducing the computational overhead associated with evaluating numerous architectures through direct training. By leveraging predictive models, this method accelerates the NAS process while maintaining high-quality architecture selection. By integrating these advanced NAS methodologies, researchers aim to strike a balance between model accuracy, computational efficiency, and scalability. These improvements pave the way for more accessible and resource-efficient deep learning architectures, enabling broader adoption of automated neural network design in practical applications [51].

The Wiener model, an important framework for representing nonlinear dynamic systems, has been widely studied in engineering applications, particularly in the domains of system identification and control. The traditional Wiener model consists of a static nonlinear block followed by a linear dynamic block, and it is often used to approximate nonlinearities in various engineering systems. However, in recent years, there has been a growing interest in extending the Wiener model using fuzzy logic systems to better handle the inherent uncertainties and complexities in real-world applications. Fuzzy Wiener models combine the Wiener structure with fuzzy inference systems (FIS), such as the Takagi–Sugeno (TS) fuzzy model, to improve the model's flexibility in approximating nonlinear systems under uncertainty. These fuzzy extensions [19] are especially useful in applications where the system's nonlinear behavior is complex and not easily captured by conventional methods. Recent advancements in fuzzy Wiener models have demonstrated their utility in various engineering applications [39]. For example, Tarek et al. [17] introduced an adaptive fuzzy Wiener model for control of nonlinear systems, where fuzzy rules were used to dynamically adapt to the nonlinear behavior of the system. This approach outperforms traditional linear models in terms of accuracy and robustness. In a similar vein, Sun and Hou [38] developed a fuzzy Wiener model for the identification and control of nonlinear mechanical systems, showcasing the model's ability to handle the nonlinearities and uncertainties commonly encountered in such systems. Moreover, Hooi et al. [41] explored fuzzy Wiener models with dynamic fuzzy rule bases, which enhance the flexibility of the model in capturing time-varying nonlinearities. This adaptation is particularly beneficial in applications such as robotics, automotive systems, and industrial processes, where the nonlinear dynamics may change over time due to varying operating conditions. These developments highlight the growing trend of using fuzzy systems to extend the classical Wiener model, providing a more adaptive and robust solution for nonlinear system identification and control in engineering fields.

The Hammerstein model, which separates the nonlinearities of a system into a static nonlinear block followed by a linear dynamic block, has long been a popular approach in system identification and control. Recently, this model has been extended using fuzzy Takagi–Sugeno–Kang (TSK) systems [54] to better handle uncertainties and nonlinearities, particularly in the context of networked and distributed systems. The fuzzy TSK-based Hammerstein model has been extensively explored for nonlinear control in networked systems, where challenges such as communication delays, packet losses, and bandwidth limitations are prevalent. In these systems, the ability to model and compensate for nonlinearities while accounting for communication constraints is crucial for maintaining high system performance. Recent studies have demonstrated the efficacy of fuzzy TSK-based Hammerstein models in such scenarios. Feng et al. [22] proposed a robust control strategy for networked systems with communication delays and packet losses, using a fuzzy TSK-based Hammerstein model to address the nonlinearities of the network dynamics. The fuzzy TSK model was found to be an effective tool in adapting to the uncertainties and nonlinearities in these systems, ensuring stable and robust control despite the presence of communication issues. Furthermore, Talpur et al. [40] introduced a hybrid approach that combines fuzzy TSK-based Hammerstein models with deep learning techniques. This approach was applied to control nonlinearities in smart grid systems, where the model's flexibility allowed it to adapt to the time-varying dynamics and uncertainties inherent in power systems. The integration of deep learning with the fuzzy TSK-based Hammerstein model proved to be highly effective in improving the system's performance under challenging conditions. In a more recent study, Tavoosi et al. [42] focused on the application of fuzzy TSK-based Hammerstein models in distributed

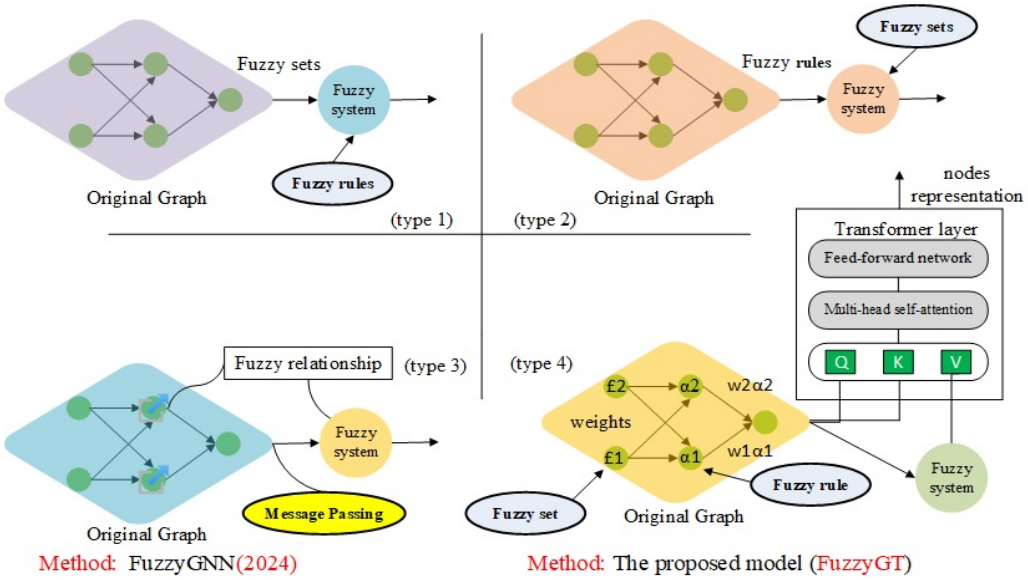


Figure 1: shows four traditional fuzzy learning systems. The combined model (CM) fits into type 1 or type 2; FGNN is categorized under type 3, while the suggested model is classified as type 4.

network systems. The study addressed communication constraints such as bandwidth limitations and latency, and demonstrated that the fuzzy TSK-based Hammerstein model could effectively mitigate the impact of these constraints while ensuring accurate control. This research underscores the utility of fuzzy TSK-based models in nonlinear control of networked systems, particularly in modern distributed environments.

Generally, a Graph Transformer architecture, with its self-attention mechanism, has a high capability to understand complex relationships between features and graph structures in a parallel and efficient manner. This capability makes the Transformer a powerful tool for extracting global graph structures. The Graph Transformer (GT), with its self-attention mechanism [36], allows for the modeling of complex relationships between graph elements and transforms the orginal graph into a target graph with a different structure while preserving semantic and structural information. The graph G is represented as an ordered pair (V, E), where V is the set of nodes and E is the set of edges. Each node v_i is represented in a d-dimensional space and indicates the features of that node. The edges e_ij have features that show the relationship between two nodes. This feature is a scalar value indicating the strength of the connection and can be generalized to more complex states where edges are vectors with higher dimensions [13].

The Graph Transformer (GTR) takes an input graph $G = (V, E)$ and transforms it into an output graph $G' = (V', E')$. It is noteworthy that the input and output graphs do not necessarily have the same structure and can exhibit different features and relationships between their nodes and edges. Useful information is extracted from the graph structure. To achieve this, the features of each node are modified and transferred to other nodes. The amount of information transferred between nodes depends on the strength of the connection between them. This process is formulated as follows:

$$X'_j = \sigma\left(X'_j + \sum_{i=1}^{n} \alpha_{ij} W^V x_i\right), \tag{1}$$

where σ is a nonlinear activation function, and $W^V \in \mathbb{R}^{d_x \times d_z}$ is the matrix of parameters. In many cases, complete information about the connections between nodes in the source and target graphs is not available. To address this challenge, an attention mechanism can be used to automatically learn the relative importance of each node in the source graph with respect to the nodes in the target graph. In this method, the connection weights between the nodes of the two graphs are treated as model parameters and are optimized during the training process. Specifically, the attention weight α_{ij} of each element x_j is calculated using a softmax function:

$$\alpha_{ij} = \frac{\exp\left(\frac{(x_i W^Q)(x_j W^K)^T}{\sqrt{d_z}}\right)}{\sum_{k=1}^{n} \exp\left(\frac{(x_i W^Q)(x_k W^K)^T}{\sqrt{d_z}}\right)}, \tag{2}$$

where α_{ij} is the attention weight of the edge connecting original node i to target node j. Here, d_z refers to the dimension of Q, and W^Q, W^K, and W^V are parameters to be learned. We then calculate the node representation to

transfer structural information to the sublayer output in the following manner:

$$z_i = \sum_{j=1}^{n} \alpha_{ij} \left(W^V x_j + W^R x_r \right), \tag{3}$$

Note that x_j represents the feature vector of node j in the input graph, x_r represents the vector for the element pair (x_i, x_j), and W^R is a parameter matrix.

Fuzzy learning plays a crucial role in deep neural networks. For instance, it can enhance the performance of deep learning models, particularly when dealing with data that is uncertain or ambiguous. Moreover, fuzzy learning techniques can be used to fine-tune parameter values within the layers of neural networks, thereby improving the model's overall effectiveness. Fuzzy systems are valuable in various applications, including pattern recognition, time series forecasting, and natural language processing, especially when the data is noisy or unclear. This integration enables deep networks to function more efficiently in complex environments [40].

In Section 1, we examined the key challenges of using Graph Neural Networks (GNNs), Graph Transformers (GTs), and fuzzy systems separately. We emphasized the importance of exploring alternative approaches to overcome the limitations of GNNs and improve their effectiveness and resilience. These alternative approaches include hybrid models that combine Graph Transformers with appropriate fuzzy systems.

In recent years, deep learning models have been employed to implement various fuzzy learning applications based on big data in graph networks. The use of fuzzy learning systems enables the handling of uncertainties and ambiguities present in real-world data [21]. A fuzzy learning approach is based on the implementation of an integrated model in which the weights in the network are updated using fuzzy logic [48]. A fuzzy model has been designed to train specific tunable parameters in a deep neural network. In addition to these instances, researchers have sought to apply fuzzy learning to tackle educational challenges associated with big data [2]. Liu [24] introduced an innovative neural learning approach that combines a transformer with a fuzzy system in the context of few-shot learning. In this study, deep learning acts as a parallel feature extractor, and fuzzy blocks are used as evaluation criteria for the learner in representing node features. This learner is designed using Gaussian-shaped fuzzy sets [32].

The use of the Transformer model, recognized in current studies as one of the leading neural network frameworks, has introduced a novel approach in this area for examining large ordered data sequences, like text in natural language processing [23]. Graph Transformers, by utilizing the self-attention mechanism, evaluate the relevance of each input component in relation to the others, thereby capturing complex relationships within the data [45]. This feature, along with the ability of Graph Transformers to extract structure and node features across the entire graph, has significantly improved the performance of models across a wide range of natural language processing tasks. Additionally, Graph Transformers have substantial potential for use in other domains, such as social network analysis, time series forecasting, and drug discovery. The high flexibility of these models in handling both structured and unstructured data makes them a powerful tool for addressing various challenges in machine learning.

Despite extensive research, fuzzy learning has not been thoroughly studied or exploited in the adaptive frameworks of Graph Transformers (GT). The challenges include the potential increase in training parameters, leading to substantial computational costs, as well as the difficulty of designing nonlinear systems that establish key relationships among predictor and outcome variables. This paper introduces a hybrid model that combines Graph Transformers (GT) with fuzzy models to process extensive datasets. The proposed model enables weight learning at the global graph level, utilizes representation sampling to minimize the number of training parameters, and uses logistic regression functions to model the relationships between inputs and outputs.

3 Methodology

3.1 Unified integration of fuzzy systems and graph transformer

This section introduces the integration of features from our proposed fuzzy model and graph transformer at both the global and local graph levels, referred to as the Fuzzy Logic-Based Graph Transformer (FuzzyGT). This integration facilitates learning feature weights by considering both the attributes and structures of nodes within FuzzyGT. It employs a mapping framework based on fuzzy logic and attention weights, consisting of two components: fuzzy logic-based representation sampling and input-output mapping. In this approach, the feature weights learned by the fuzzy model are combined with the graph transformer architecture to generate the final node representation in the graph. The proposed structure of FuzzyGT is shown in (See Figure 2). The input to FuzzyGT consists of the feature vectors of graph nodes, which are transferred to the next layer through positional encoding. The primary representation of each node is an $\hat{D}$-dimensional vector, denoted as $X = (x_1, x_2, \ldots, x_{\hat{D}})$. The global and local information of the node in

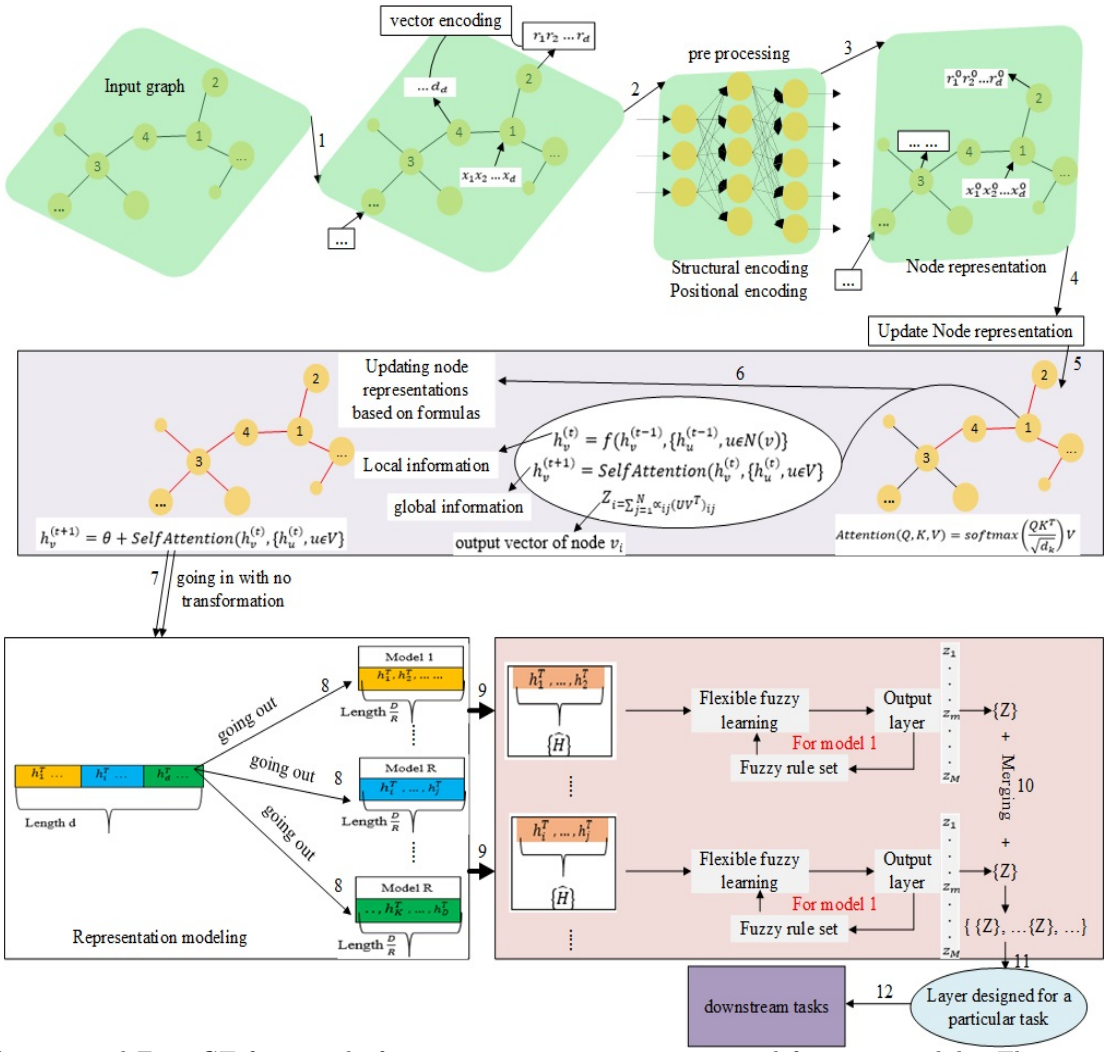


Figure 2: The proposed FuzzyGT framework, featuring its primary component and fuzzy set module. The vector $h_1^T, \ldots, h_D^T$ represents the node encoding in a hidden layer, where $\{\hat{Z}\}$ indicates the modeled node features (a subset of the node encoding). The output vector $Z = (Z_1, \ldots, Z_m)$ is generated through the fuzzy output layer for an individual model.

the graph is updated using positional and structural encoding, ultimately leveraging all features and structures of the graph. This results in more efficient expressiveness and improved node representation. The representation of the d-th node in the initial phase is given by $Z_V^0 = (x_1^0, x_2^0, \ldots, v^0)$, where $v = 1, 2, \ldots, V$, and V refers to the number of nodes in the input graph. The Graph Transformer updates the node representations based on attention to other nodes and local information $h_v^{(t)}$ and global information $h_v^{(t+1)}$ in the graph. The output of the GT layers is denoted by $h_v^{(t+1)}$, representing the v-th node in the self-attention layer. Then, this node representation is updated using the multi-head self-attention layer based on a fuzzy forest, which will be explained thoroughly in the following subsection. Finally, the feed-forward layer processes the final node representation to produce the output.

The core idea behind the Graph Transformer (GT) is to learn node representations using an attention mechanism, ensuring the preservation of both local and global information from the graph's structure and features [7]. This process is referred to as representation learning, where the node embedding is considered the key feature of the node. There are two main challenges in integrating fuzzy systems with GT: (1) The issue of feature space, resulting in an exponential growth in fuzzy conditions. (2) The possible inaccuracies in modeling the relationships between inputs and outputs. To address the first challenge, long representations are divided into smaller sub-vectors, which significantly reduces the quantity of fuzzy rules required. To address the second challenge, various regression functions are used to improve the modeling of the relationship between inputs and outputs. This approach of integrating fuzzy systems is applied to both node and edge embedding learning. The following sections will delve into the integrated learning approach, the proposed strategies to resolve these challenges, and the detailed process.

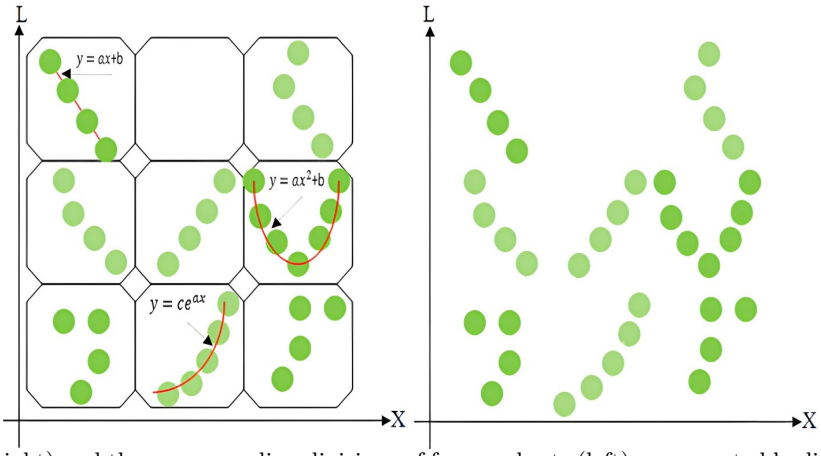


Figure 3: Modeled data (right) and the corresponding divisions of fuzzy subsets (left), represented by linear, nonlinear, quadratic, or exponential functions.

3.2 Fuzzy learning rule

Learning fuzzy systems simplifies the output function by linearly dividing the input space into subspaces. For example, in (See Figure 3) (left), establishing a function between variables R and D is challenging due to varying relationships across regions. The data exhibit linear, quadratic, and exponential trends, making a single function impractical. Overlapping fuzzy subsets further complicate direct modeling. In contrast, Figure 3 (right) shows how fuzzy logic learning transforms the input space, applying appropriate functions to different regions. The fuzzy logic learning function works similarly to a transformation function, mapping the output variables into a different feature space for further analysis. This approach simplifies complexity and makes the data more interpretable. The key component of this transformation is the relationship $X = F(L)$ for the dataset (X_t, l_t), which represents features and labels. To train the fuzzy learning system, the least squares estimation rule is used to optimize or minimize the membership degree error μ, which is defined as follows:

$$\mu(A, B, X_t, l_t) = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(AX_t + B - l_t)^2}, \tag{4}$$

Here, X_t denotes the location at time t in the target space, and $f(X_t)$ is the approximated computed result. The value l_t represents the data that has been actually observed. Parameters A and B serve as weights or optimized values for better fitting the data. The goal of curve fitting aims to reduce μ by taking into account the characteristics of A and B.

$$\min \mu(A, B, X_t, l_t) = \min \sum_{k=1}^{n} \left[2(X_t + B)^T l_t + l_t^T l_t - (AX_t + B)^T (AX_t + B)\right], \tag{5}$$

We rewrite the above formula as follows:

$$\begin{cases} \sum X_t l_t = BA^2 \sum X_t \\ \sum l_t = nB + A \sum X_t \end{cases}$$

This is a common challenge in solving systems of linear equations, where A and B are two unknowns which define the optimal weights for curve fitting and accurately solve the equations. The goal is to minimize the value of μ in Equation (4).

3.3 Fuzzy representation modeling

The goal of introducing the representation modeling approach is to reduce the quantity of fuzzy rules, thereby decreasing the amount of adjustable parameters. This model operates after the Graph Transformer layers on the GT final node embedding, denoted as $h_v^{(t+1)}$. Next, $h_v^{(t+1)}$ is fed as the input into the embedding sampling model. The representation modeling process consists of two steps, specifically beginning with representation sampling. First, consider a scenario where the node representation is updated as a whole, without partitioning into different subsets. For simplicity, consider a two-dimensional representation vector, with each axis (dimension) defined by two fuzzy sets. In this case, four fuzzy rules are generated, partitioning the two-dimensional input space into four regions with flexible (fuzzy) boundaries [28]. In general, assuming the input is a d-dimensional graph represented as $Z_V^0 = (x_1^0, x_2^0, \ldots, v^0)$, and the quantity

of fuzzy sets in all dimensions is defined as N (manually specified), then N^d fuzzy rules are required to partition the d-dimensional feature space into N^d subregions. Since each fuzzy rule covers only one subregion, the "curse of dimensionality" arises, which is a major factor contributing to computational complexity. Current research specifically addresses this challenge, as noted in studies like [15]. However, the proposed method relies on adapting fuzzy rules by adjusting weight parameters, but implementing this approach is highly challenging. The inference process in a fuzzy rule can be described as a weighted sum of the activation function and the fuzzy logic associated with the outcome, which is often a linear function. Therefore, for each fuzzy rule, $d+1$ parameters are typically required. If the output is a multidimensional vector, the final output is represented as $Z = (Z_1, \ldots, Z_m)$, where m is a fixed parameter that defines the dimensionality of the output vector based on the complexity of the problem (See Figure 2 for more details). Consequently, the total number of parameters in the output layer is calculated as $N^d \times (d+1) \times m$. This value determines the computational cost of the FuzzyGT model. To address the issue of parameter growth in fuzzy learning models, a multi-head attention layer based on a fuzzy forest is utilized, where the representation of each node in the graph is derived from the set of attention heads. Suppose the representation of a node is given as $Z_V^0 = (x_1^0, x_2^0, \ldots, v^0)$. In the multi-head attention layer, this representation is divided as follows:

$$\begin{aligned}
\text{Section } 1 &= \{h_1^T, h_2^T, \ldots, h_{D/R}^T\},\\
\text{Section } 2 &= \{h_{D/R+1}^T, h_{D/R+2}^T, \ldots, h_{2D/R}^T\},\\
&\vdots\\
\text{Section } R &= \{h_{D-D/R+1}^T, \ldots, h_{D-1}^T, h_D^T\}.
\end{aligned}$$

Here, R is chosen such that $\frac{D}{R} \in \mathbb{Z}$ is an integer.

3.4 Proposed fuzzy learning module (FuzzyGT framework)

This module enhances the feature learning process by combining the learned features in the input graph through the fuzzy model with components present in the graph transformer. The fuzzy graph transformer module transforms each of the input features into a vector $X = (x_1, x_2, \ldots, x_{\hat{D}})$. inally, all output embedding vectors are concatenated into a single vector represented as $h_v^{(t+1)}$ (See Figure 4). First, the input vector $X = (x_1, x_2, \ldots, x_{\hat{D}})$ from the sampling stage is passed to the attention layer through positional encoding and structural encoding. In this layer, each element $x_{\hat{D}}$ is mapped to a membership degree using the fuzzy set $\tilde{\mu}_m^d$. Here, m represents the count of fuzzy sets for each axis, while d denotes the dimensionality of the input data for the graph transformer, with both specified as defined parameters. The fuzzy set $\tilde{\mu}_m^d$ is defined as follows:

$$\tilde{\mu}_m^d(\tilde{x}_d) = \exp\left(\frac{1}{\sqrt{2\pi(\tilde{\rho}_n^d)^2}}\right), \tag{6}$$

Where $d = 1, 2, \ldots, D$ and $n = 1, \ldots, N$, $\tilde{\mu}_m^d(\tilde{x}_d)$ represents the fuzzy membership function associated with the m-th fuzzy set in the d-th dimension. $\tilde{x}_d$ is the transformed input variable in the d-th dimension, and $\tilde{\rho}_n^d$ denotes a parameter that influences the spread of the fuzzy set.

Secondly, the rule aggregation layer of the fuzzy system is responsible for determining the relationship between the input and output within each "subregion". This subregion is defined based on a specific d-dimensional integration of fuzzy sets, derived from their max-product combination. The task of the fuzzy rule integration layer is to estimate the relationship between the input variables $X = (x_1, x_2, \ldots, x_{\hat{D}})$ and the integration function $f_c^l = F(\tilde{Q}_c^l, X, h_c^l)$. Here, f_c^l represents a fuzzy rule, where $\tilde{Q}_c^l = (\tilde{Q}_{1,c}^l, \ldots, \tilde{Q}_{d,c}^l)$ is defined. The function F is expressed as follows:

$$F = \sum_{i=1}^{n} Z_i \cdot (G_i X_i), \tag{7}$$

$$Z = (\tilde{Q}_{1,c}^l X_1, \ldots, \tilde{Q}_{\tilde{d},c}^l X_{\tilde{d}}, \ldots, \tilde{Q}_{\tilde{D},c}^l X_{\tilde{D}}, h_c^l), \tag{8}$$

where Z_i represents the components of Z, X_i denotes the components of the vector X, and G_i represents the components of the vector G. Note that a simple linear function, defined as $F = ZI$, is not suitable for capturing complex nonlinear relationships between inputs and outputs. Linear and logistic regressions hold true within very small subsets of fuzzy rules, and the overall output exhibits nonlinear behavior. However, this process requires excessive subdivision of subsets and an increase in the number of rules, which contradicts the goal of reducing complexity. It

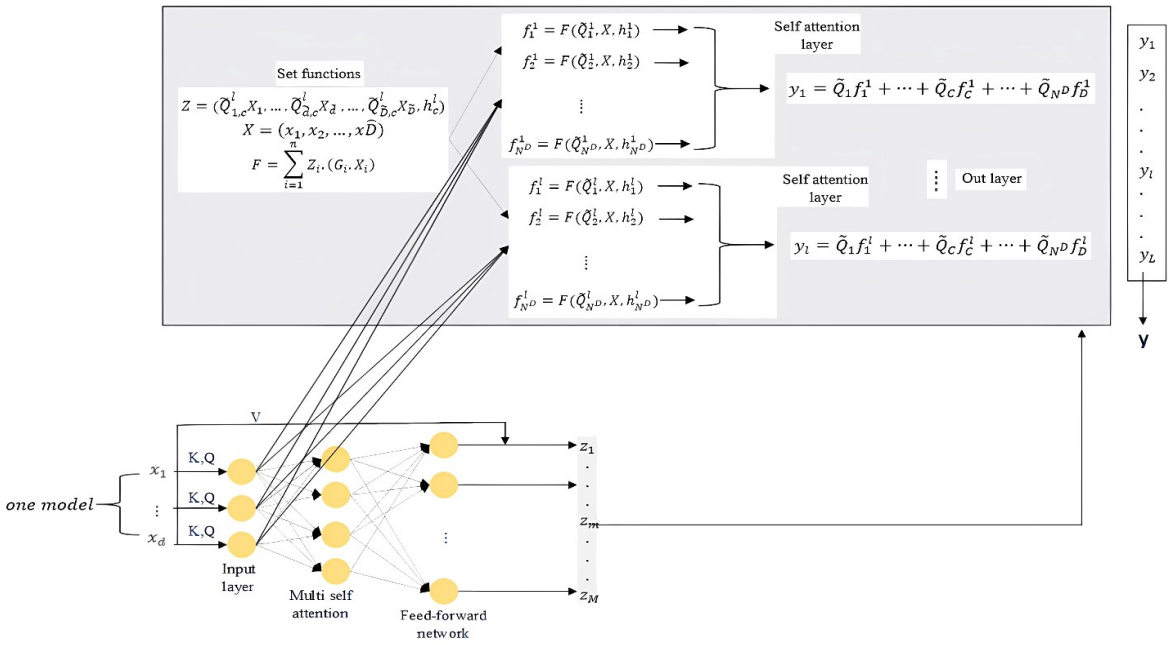


Figure 4: Graph Transformer-Fuzzy Convergence. This module employs a Transformer-Fuzzy model that incorporates a tunable polynomial function to capture and model the connection between input and output variables in the final output layer.

is important to highlight that when using a graph transformer to process an input graph, the dimensionality of the problem is significantly higher than that of standard regression tasks. Therefore, integration and sampling methods are employed prior to processing. Therefore, integration and sampling methods are employed prior to processing. In general, the result of the c-th rule is represented as $f_c^l = F(\tilde{Q}_c^l, X, h_c^l)$, and the overall count of trainable parameters is given by $\tilde{c}^d \times (\tilde{d}+1)$. Controlling the number of fuzzy rules plays a crucial role in reducing the complexity of the graph transformer, and the trainable parameters also help balance the fuzzy rules.

4 Experiments and results

In this section, four diverse datasets from the field of machine learning are tested to evaluate the performance of the proposed FuzzyGT model and compare it with state-of-the-art (SOTA) methods.

4.1 Implementation details

We utilized three well-known graph datasets: Cora [27], Citeseer [5], and Pubmed [8], which are categorized as citation networks. In these datasets, each node represents a document, and each edge indicates a citation link between two documents. Each node is assigned a class label corresponding to the main topic of the respective document. Table 1 presents the statistical details of these datasets. For data splitting, standard methods were employed to allocate data into training, validation, and testing sets. The experiments were conducted 10 times using fixed parameter settings and the same random seed. The overall performance was assessed by calculating the average (Avg.) and standard deviation (Sd.) across these 10 runs. The parameter size limit for the Cora, Citeseer , and Pubmed datasets was set to 500,000. The Adamic Adar optimizer [31] was used in all experiments for learning rate adjustment with default settings. The values of β_1, β_2, and ε were set to 0.89, 0.899, and 1×10^{-9}, respectively. The learning rate was initialized at 0.002, with the weight decay configured at 1×10^{-8}. The learning rate was kept constant in the early stages and then gradually decayed following a cosine schedule.

The maximum number of epochs was set to 1000 for Citeseer and 90 for the other datasets. The batch size was configured to 32 for Cora and Citeseer and 16 for Pubmed. All experiments were executed on the Google Colab platform, utilizing one NVIDIA GeForce GPU with 6 GB of memory and two V100-PCIE GPUs with 16 GB of memory each. FuzzyGT also includes two sampling methods: Gaussian and Stratified. For each dataset, we evaluate three types of F consequent functions-first-order, mixed-order, and second-order-resulting in six distinct combinations: Gaussian-first-order (G(0)), Gaussian-second-order (G(1)), Gaussian-mixed-order (G(0.5)), Stratified-first-order (S(0)), Stratified-second-order (S(1)), and Stratified-mixed-order (S(0.5)). Table 1 presents the statistics for these three datasets. Avg.W represents the average number of words per node. The parameters related to the training of GT are listed in Table 2.

Table 1: Overview of the dataset statistics.

Dataset	nodes	edges	features	classes	Avg.W
Cora	2,708	5,069	1,433	7	18
Citeseer	3,327	3,668	3,703	6	31
Pubmed	19,717	44,338	500	3	50

Table 2: FuzzyGT hyperparameters for three machine learning datasets.

Hyper-parameters	Cora	Citeseer	Pubmed
Embed. dim.	64	64	32
Self-attn. heads	4	4	4
Weight decay	1e-8	1e-8	1e-8
Learning rate	0.002	0.002	0.002
Gradient clip norm	1.0	1.0	5
LR scheduler	cosine,warm-up	cosine,warm-up	constant
Batch size	32	32	16
# Sample (R)	10	12	16
L	2	2	2
C	3	3	3

Table 3: Results on test data across five datasets is presented as the average and standard deviation from 10 runs. The best results are highlighted in bold, and the second-best results are marked in italics. FuzzyGT delivers the best performance by utilizing different continuous functions F.

Model	**Cora**		**Citeseer**		**Pubmed**	
	Accuracy↑	**MAE↓**	**Accuracy↑**	**MAE↓**	**Accuracy↑**	**MAE↓**
GCN	*79.68 ± 0.15*	*75.31 ± 0.06*	*65.24 ± 0.36*	*0.36 ± 0.10*	*0.39 ± 0.16*	*0.41 ± 0.041*
GAT	*83.40 ± 0.03*	*83.50 ± 0.01*	*84.53 ± 0.27*	*0.38 ± 0.25*	*0.31 ± 0.20*	*0.30 ± 0.031*
M-Graphormer	*86.72 ± 0.27*	*85.80 ± 0.06*	*86.61 ± 0.33*	*0.10 ± 0.031*	*0.11 ± 0.008*	*0.13 ± 0.040*
GTN	*89.60 ± 0.39*	*89.92 ± 0.04*	*90.70 ± 0.14*	*0.09 ± 0.005*	*0.08 ± 0.003*	*0.08 ± 0.052*
FuzzyGNN	*90.10 ± 0.23*	*90.12 ± 0.03*	*90.15 ± 0.23*	*0.06 ± 0.004*	*0.06 ± 0.021*	*0.06 ± 0.060*
GRAPH-BERT	*91.85 ± 0.05*	*91.74 ± 0.05*	*90.80 ± 0.44*	*0.13 ± 0.001*	*0.14 ± 0.032*	*0.15 ± 0.030*
GRAPHSAGE	*80.46 ± 0.04*	*78.41 ± 0.24*	*75.55 ± 0.55*	*0.29 ± 0.005*	*0.30 ± 0.002*	*0.31 ± 0.008*
FuzzyGT	**97.79 ± 0.22**	**98.56 ± 0.02**	**99.01 ± 0.22**	**0.04 ± 0.002**	**0.04 ± 0.001**	**0.03 ± 0.004**

The embedding dimensions, global attention heads in the graph transformer [18], and other parameters are specified. The sample size (R) varies across datasets, while the output vector dimension (L) is fixed at 2. Additionally, the count of fuzzy sets applied at each stage is consistently 3 across all datasets.

To better understand the influence of hyperparameters on the performance of the Graph Transformer (GT) model, we conduct a parameter sensitivity analysis. This analysis evaluates the impact of several key parameters, including embedding dimension, the number of attention heads, sample size (R), learning rate, and others, on model performance across different datasets. As outlined in Table 2, the hyperparameters for training the GT model vary across the Cora, Citeseer, and Pubmed datasets. By systematically varying one parameter at a time while keeping others fixed, we assess how these changes affect accuracy, training time, and overall model efficiency. Specifically, we explore how different embedding dimensions and attention heads influence the model's ability to capture graph structure and the trade-off between model complexity and computational cost. We also examine the effect of sample size (R), which differs across datasets, on the model's generalization ability, particularly in terms of how well it performs on unseen data. In addition, we investigate the role of the learning rate and gradient clipping norm in stabilizing training and speeding up convergence. Other factors, such as weight decay, batch size, and learning rate scheduling, are also tested to understand their impact on training dynamics and final model performance. This sensitivity analysis will provide valuable insights into how each hyperparameter contributes to the overall success of the model, offering guidance for future hyperparameter tuning and optimization across various graph-based tasks.

4.2 Comprehensive results and analysis of the large dataset

In this study, we compare FuzzyGT with several well-known GNN models and advanced methods, including GCN [55], GAT [43], M-Graphormer [6], GTN [52], FuzzyGNN [37], GRAPH-BERT [56], and GRAPHSAGE [47]. The results of the optimal performance of FuzzyGT across six different settings (G(0), G(0.5), G(1), S(0), S(0.5), S(1)) are presented in Table 3. Each of these experiments was run independently ten times, and the mean and standard deviation of the results are also provided.

In Table 3, we compare the performance of various models on five datasets, reporting both *accuracy* (higher is better) and *mean absolute error (MAE*, lower is better). Each result is averaged over 10 independent runs, with the

Table 4: Performance on test data across three datasets using FuzzyGT.

Method	**Cora**		**Citeseer**		**Pubmed**	
	Accuracy↑	**MAE↓**	**Accuracy↑**	**MAE↓**	**Accuracy↑**	**MAE↓**
FuzzyGT (G(0))	97.59 ± 0.19	0.07 ± 0.006	**98.56 ± 0.03**	0.09 ± 0.023	89.99 ± 0.01	0.11 ± 0.008
FuzzyGT (G(0.5))	**97.79 ± 0.22**	0.05 ± 0.10	98.53 ± 0.04	0.05 ± 0.014	99.00 ± 0.07	0.10 ± 0.001
FuzzyGT (G(1))	97.60 ± 0.63	**0.04 ± 0.002**	98.50 ± 0.06	**0.04 ± 0.002**	**99.01 ± 0.20**	0.08 ± 0.005
FuzzyGT (S(0))	97.64 ± 0.28	0.08 ± 0.012	98.49 ± 0.05	0.07 ± 0.001	89.97 ± 0.27	**0.03 ± 0.002**
FuzzyGT (S(0.5))	97.70 ± 0.08	0.09 ± 0.020	98.45 ± 0.08	0.10 ± 0.008	89.95 ± 0.08	0.05 ± 0.001
FuzzyGT (S(1))	97.69 ± 0.54	0.10 ± 0.007	98.41 ± 0.07	0.12 ± 0.007	89.92 ± 0.30	0.03 ± 0.004

Table 5: Performance on test data across three datasets using FuzzyGT. The values indicate average runtime per period and total overall time.

Data	**FuzzyGT**					
	G(0.5)	G(0)	G(1)	S(0.5)	S(0)	S(1)
Cora	268.24 s / 5.28 h	268.32 s / 5.12 h	268.30 s / 5.27 h	268.29 s / 5.28 h	268.31 s / 5.69 h	268.30 s / 5.69 h
Citeseer	289.27 s / 6.47 h	289.27 s / 6.40 h	289.27 s / 6.31 h	289.27 s / 6.30 h	289.27 s / 6.29 h	289.27 s / 6.25 h
Pubmed	320.14 s / 7.25 h	320.14 s / 7.10 h	320.14 s / 7.02 h	320.14 s / 7.05 h	320.14 s / 7.01 h	320.14 s / 7.00 h

best outcomes highlighted in bold and the second-best shown in italics. FuzzyGT excels in all metrics, achieving impressive accuracy values such as 97.79% on *Cora*, 98.56% on *Citeseer*, and 99.01% on *Pubmed*, while also yielding the lowest MAE in each case. Furthermore, we conducted six detailed experiments using different F functions within the FuzzyGT model across three different datasets. All experiments were executed independently using the same experimental settings. Table 4 presents the results for the various configurations $G(0)$, $G(0.5)$, $G(1)$, $S(0)$, $S(0.5)$, and $S(1)$ of FuzzyGT, where each dataset was tested independently ten times. According to the results in Table 4, for the *Citeseer* dataset, the $G(0)$ configuration achieves the highest performance, followed by $G(0.5)$ and $G(1)$ as the second and third best, respectively. In the *Cora* dataset, the Gaussian-based configuration $G(0.5)$ yields the highest accuracy, with the stratified configurations $S(0.5)$ and $S(1)$ ranking next. Finally, for the *Pubmed* dataset, the $G(1)$ configuration with a specific parameter achieves the best performance. Overall, regression-based functions tend to perform slightly better than other configurations in most cases.

Table 5 presents an analysis of the computational complexity associated with the experiments. All experiments were conducted using the same hardware configuration. The times reported in the table represent the average duration from ten repeated runs of the experiments. Based on the results in Table 5, it can be concluded that there is no significant difference in the computational time required for the various F functions.

4.3 Ablation study

In this section, we perform an ablation study to compare the performance of the baseline model with a new fuzzy transformer-based model. The baseline model is a transformer network, which serves as part of the proposed FuzzyGT module. In machine learning and classification tasks, accuracy is a key metric used to assess how well the model is performing. Simply put, accuracy is the proportion of predictions that the model gets right. The formula for calculating accuracy is as follows:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}.$$

Precision refers to the proportion of true positives among all the positive predictions made by the model. Recall, on the other hand, indicates how well the model identifies all the actual positives. Overall classification accuracy measures the fraction of samples that were correctly classified out of the total samples.

$$\text{Precision} = \frac{TP}{TP + FP}.$$

$$\text{Recall} = \frac{TP}{TP + FN}.$$

To calculate the Micro F1 score, we first need to compute Micro-Precision and Micro-Recall. The concept of the Micro average is to treat all data points equally, without taking into account any potential class differences. As a result, Micro-Precision is calculated in the following way:

$$\text{Micro Average Precision} = \frac{\sum_{k=1}^{K} TP_k}{\sum_{k=1}^{K} \text{Total Column}_k} = \frac{\sum_{k=1}^{K} TP_k}{\text{Grand Total}}.$$

Table 6: Attributes of Graph Datasets.

Dataset	Number of Graphs	Class Labels	Avg. Nodes	Avg. Edges	Evaluation Metric
Amazon	1500	10	25.08	44.58	accuracy
Facebook	5000	5	20.45	53.30	accuracy
PTC-MR	344	2	14.33	34.08	accuracy
IMDB-BINARY	1000	2	19.77	96.53	accuracy

Table 7: The findings from the Ablation Study conducted across five different datasets, using the base AGT model and the AGT model enhanced with the proposed fuzzy module, AGAT+F.

Name	Models	Accuracy	Precision	Recall	Micro-F1	AUC	Runtime (Seconds per Epoch)
Amazon	AGT	72.12	58.85	49.32	37.26	84.65	-
	AGT+F	**86.31**	**69.62**	**60.36**	**46.01**	**87.18**	118.34
Facebook	AGT	64.01	48.60	46.87	40.50	70.42	-
	AGT+F	**71.69**	**56.14**	**55.10**	**54.31**	**73.41**	203.56
PTC-MR	AGT	75.45	60.28	59.43	53.47	85.32	-
	AGT+F	**88.12**	**72.31**	**70.22**	**67.33**	**88.20**	82.71
IMDB-BINARY	AGT	78.32	50.73	50.30	32.48	79.70	-
	AGT+F	**89.24**	**58.50**	**65.31**	**42.13**	**84.15**	145.22

When evaluating Micro-Recall, we notice that its value is identical to that of Micro-Precision. This is because the calculation of both metrics is based on the total number of true positives. As a result, Micro-Recall is calculated in the following way:

$$\text{Micro Average Recall} = \frac{\sum_{k=1}^{K} TP_k}{\sum_{k=1}^{K} \text{Total Rows}_k} = \frac{\sum_{k=1}^{K} TP_k}{\text{Grand Total}}.$$

So, Micro-F1 is calculated in the following way:

$$\text{Micro-F1} = \frac{\sum_{k=1}^{K} TP_k}{\text{Grand Total}}.$$

The transformer model starts off with an attention layer that leverages Laplace positional encoding. From there, it moves on to a Multi-Head Attention layer and wraps up with a task-specific Feedforward layer. This basic structure forms the foundation of a three-layer network, known as the Transformer Ablation Graph (AGT). In the enhanced version, a fuzzy inference module is inserted just before the task-specific layer, turning the model into a four-layer version called AblationGT-fuzzy (AGT+F). For both models, the experimental settings are identical. In this configuration, the positional encoding is configured to 4, the ReLU activation function is applied, and the feature node count for the test datasets is 32 and 64. The dropout rate for the attention layer is set at 0.2, and the transformer layer uses four heads. Adam is the chosen optimization algorithm, with a base learning rate of 0.004, and training is limited to a maximum of 80 epochs. Large datasets collected from social networks and artificial intelligence domains for tasks involving graph classification and regression are thoroughly described in Table 6. The experiments are independently conducted twice using AGT, and the results are evaluated for accuracy. Additionally, datasets with two or more classes are selected to facilitate the measurement of metrics such as micro-F1, precision, and recall.

Table 7 presents the results of an ablation study conducted on four distinct datasets. In this analysis, we compared the performance of the base model, AGT, with its enhanced version, AGAT+F, which integrates a fuzzy inference module we proposed. The findings demonstrate that, overall, AGAT+F provides superior performance in terms of average accuracy across all datasets compared to AGT. The largest accuracy improvement of 14.19% was observed on the Amazon dataset, while the smallest increase of 7.68% was seen on the IMDB-BINARY dataset. Specifically, accuracy improvements of 14.19%, 7.68%, and 12.67% were achieved on the Amazon, PTC-MR, and Facebook datasets, respectively. In addition to this, the recall metric showed notable improvements, with increases of 10.04%, 10.79%, and 8.23% on the Amazon, PTC-MR, and Facebook datasets. Furthermore, the F1 and AUC scores also saw significant gains across all four datasets. These results highlight the effectiveness of the proposed fuzzy module in enhancing the model's overall performance across various evaluation metrics.

Our experiments with moderately large datasets (IMDB-BINARY, PTC-MR, Facebook, Amazon, e.g.) indicate that the FuzzyGT model scales well with an increase in graph size, maintaining performance without a proportional increase in computational cost. The model leverages optimized sampling techniques that ensure efficient handling of graph data, allowing for practical training times even as graph complexity increases. However, for applications involving extremely large graphs, such as those found in social media networks or financial datasets, we acknowledge that additional strategies may be necessary. Specifically, hierarchical or distributed computing frameworks could be employed to further enhance scalability. Techniques such as graph partitioning, distributed message passing, and mini-batch training could allow for efficient processing of large-scale graphs by distributing the computational load across multiple machines or processing units. These approaches could significantly improve runtime and memory efficiency,

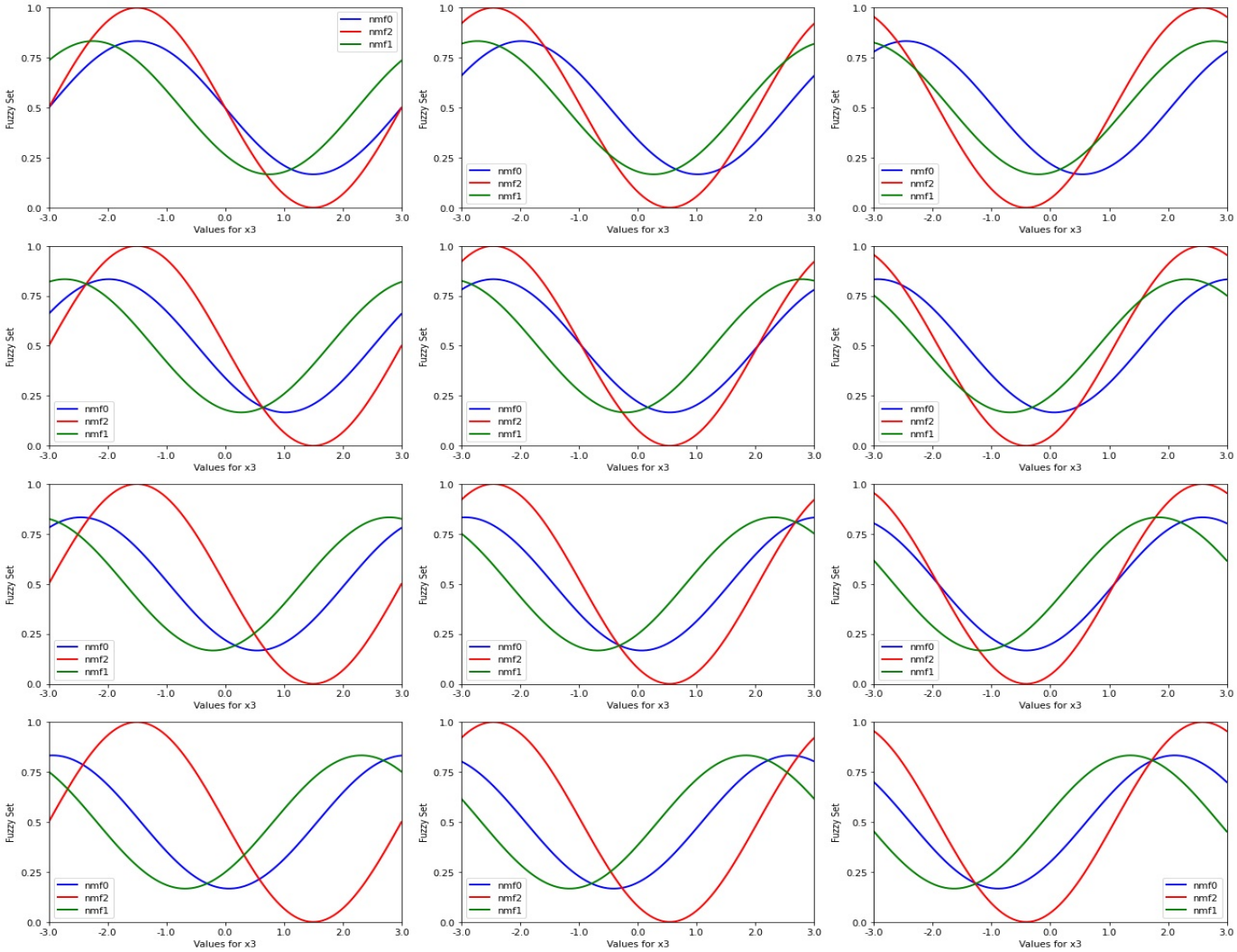


Figure 5: a) Cora b) Citseer C) Punmed. The fuzzy sets created based on the training data for each dataset in Table 4 refer to four different attributes for a data point in each subfigure, where x3 represents the number of these fuzzy sets

enabling the application of FuzzyGT to real-world large-scale networks. To validate the scalability of FuzzyGT, we conducted preliminary runtime analyses across various datasets. Our results show that the model's performance, in terms of both accuracy and runtime, remains stable as the number of nodes and edges increases, thanks to the use of efficient attention mechanisms and sampling strategies. In future work, we plan to explore further optimizations such as distributed computing frameworks, which could facilitate the model's application to graphs with billions of nodes and edges. The accuracy remains stable across different datasets as the graph size increases, demonstrating that the model's performance does not degrade as graph complexity grows. Runtime increases moderately with the size of the graph, which aligns with expectations due to the greater number of nodes and edges. However, the increase is not exponential, thanks to the efficient attention mechanisms and sampling strategies employed by FuzzyGT. The model's scalability indicates its potential applicability to larger real-world graphs in domains such as social media and financial networks.

Fuzzy logic is chosen over Bayesian methods, attention-based noise filtering, and dropout variants due to its superior ability to handle uncertainty in graph-based learning. Unlike Bayesian approaches that require predefined probability distributions, fuzzy logic adapts dynamically to noisy or incomplete data. While dropout and attention-based noise filtering improve regularization, they do not explicitly model uncertainty. Our ablation study (Table 7) confirms that integrating fuzzy logic into the Graph Transformer framework (FuzzyGT) significantly improves accuracy and recall across datasets. This highlights its effectiveness in enhancing robustness and interpretability. Future work will explore hybrid methods combining fuzzy logic with other regularization techniques.

The processed fuzzy sets are presented in Figure 5. Two key points related to this figure are as follows:

1. Figure 5 illustrates the fuzzy sets that have been trained for all inference modules R across each individual dataset.

2. For each fuzzy set, we have used an input variable transformed by the output of the transformer layers, and we have applied the same domain of $[-3, 3]$ for all variables, which results in the similar graphs seen in Figure 5.

5 Conclusion and future work

In this paper, we present a novel approach that combines fuzzy systems with graph transformers, leading to the creation of the Fuzzy Graph Transformer (FuzzyGT). Our model tackles critical challenges faced by graph neural networks (GNNs), such as over-smoothing and sensitivity to noisy or incomplete data, by harnessing the strengths of fuzzy systems within a global graph-level framework. By incorporating fuzzy logic, we enable our model to capture both local and global graph patterns more effectively, improving node representations and enhancing the overall performance of the model. We propose a framework where fuzzy learning enhances graph representations at both local and global levels. By integrating fuzzy rules and using efficient sampling techniques, we reduce the computational complexity while increasing the scalability of the model. Moreover, the addition of a fuzzy consequence layer within each subspace allows for more accurate modeling of feature relationships, thus improving the model's predictive power. Our experimental results, based on five widely recognized datasets, show that FuzzyGT significantly outperforms existing state-of-the-art models. The model demonstrates considerable improvements in predictive accuracy and robustness, particularly when working with noisy or incomplete data. These results validate the effectiveness of our integrated approach and establish FuzzyGT as a leading method in the field of graph-based deep learning. In conclusion, this work makes a meaningful contribution to the area of graph representation learning by introducing a hybrid model that seamlessly combines fuzzy logic with graph transformers. The FuzzyGT model provides a powerful solution for large-scale data analysis, opening the door to more efficient and reliable applications in graph-based machine learning. While the proposed Fuzzy Graph Transformer (FuzzyGT) represents a significant step forward in integrating fuzzy systems with graph transformers, several opportunities for future exploration remain:

- **Complex Graph Structures:** Expanding FuzzyGT to handle dynamic or temporal graphs, where node and edge representations evolve over time, would enhance its adaptability to changing graph dynamics.
- **Multi-modal Data:** Integrating multi-modal data (e.g., images, text, and graphs) into the FuzzyGT framework could improve its ability to capture complex relationships across different data types.
- **Fuzzy Systems Optimization:** Optimizing fuzzy rule generation techniques and exploring methods like evolutionary algorithms could help handle large-scale datasets more efficiently and dynamically adapt fuzzy logic parameters.
- **Scalability on Larger Graphs:** Testing FuzzyGT on large-scale graphs with millions of nodes and edges will be crucial for real-world industrial applications, ensuring the model can maintain performance at scale.
- **Interpretability:** Enhancing the interpretability of fuzzy rules within the graph transformer could make the model more transparent and accessible, especially for practitioners needing explainable AI.
- While FuzzyGT shows great potential, there are several limitations that need to be addressed in future work:
- **Computational Overhead:** The integration of fuzzy systems with graph transformers increases computational complexity, especially with large graphs. Further optimizations are needed for scalability.
- **Static Graph Assumption:** FuzzyGT currently assumes fixed graph structures, which may not be suitable for dynamic graphs in applications like social networks. Future work could explore dynamic graph handling.
- **Hyperparameter Sensitivity:** FuzzyGT is sensitive to hyperparameter selection, requiring further fine-tuning for optimal performance. Automation of this process could improve efficiency.
- **Real-World Application:** While FuzzyGT performs well on benchmark datasets, its real-world applicability, especially in noisy or incomplete data environments, requires further evaluation.
- **Rule Generation Complexity:** Generating fuzzy rules for large datasets remains computationally expensive. Streamlining this process could improve scalability and efficiency.

References

[1] M. Akram, A. Khan, A. Borumand Saeid, *Complex Pythagorean Dombi fuzzy operators using aggregation operators and their decision making*, Expert Systems, **38**(2) (2021), e12626. https://doi.org/10.1111/exsy.12626

[2] M. Alateeq, W. Pedrycz, *Logic-oriented fuzzy neural networks: A survey*, Expert Systems with Applications, **257** (2024), 125120. https://doi.org/10.1016/j.eswa.2024.125120

[3] S. Bastami, A. Abdollahpouri, R. Pir Mohammadiani, *FARW: A feature-aware random walk for node classification*, Journal of Innovations in Computer Science and Engineering JICSE, no. Online First, **1**(2) (2025). https://doi.org/10.48308/jicse.2025.237378.1039

[4] E. Bastami, et al., *A gravitation-based link prediction approach in social networks*, Swarm and Evolutionary Computation, **44** (2019), 176-186. https://doi.org/10.1016/j.swevo.2018.03.001

[5] K. D. Bollacker, S. Lawrence, C. L. Giles, *CiteSeer: An autonous web agent for automatic retrieval and identification of interesting publications*, in Proceedings of the Second International Conference on Autonomous Agents - AGENTS '98, Minneapolis, Minnesota, United States: ACM Press, (1998), 116-123. https://doi.org/10.1145/280765.280786

[6] X. Chang, J. Wang, M. Wen, Y. Wang, Y. Huang, *M-Graphormer: Multi-channel graph transformer for node representation learning*, IEEE Transactions on Big Data, (2024), 1-13. https://doi.org/10.1109/TBDATA.2024.3489418

[7] J. Chen, G. Li, J. E. Hopcroft, K. He, *SignGT: Signed attention-based graph transformer for graph representation learning*, arXiv, (2023). https://doi.org/10.48550/ARXIV.2310.11025

[8] F. Dernoncourt, J. Y. Lee, *PubMed 200k RCT: A dataset for sequential sentence classification in medical abstracts*, arXiv, (2017). https://doi.org/10.48550/ARXIV.1710.06071

[9] M. B. Dowlatshahi, V. Derhami, H. Nezamabadipour, *Fuzzy particle swarm optimization with nearest-better neighborhood for multimodal optimization*, Iranian Journal of Fuzzy Systems, **17**(4) (2020), 7-24. https://doi.org/10.22111/ijfs.2020.5403

[10] M. B. Dowlatshahi, A. Hashemi, *Unsupervised feature selection: A fuzzy multi-criteria decision-making approach*, Iranian Journal of Fuzzy Systems, **20**(7) (2023), 55-70. https://doi.org/10.22111/ijfs.2023.7630

[11] F. Fanian, M. Kuchaki Rafsanjani, *CFMCRS: Calibration fuzzy-metaheuristic clustering routing scheme simultaneous in on-demand WRSNs for sustainable smart city*, Expert Systems with Applications, **211** (2023), 118619. https://doi.org/10.1016/j.eswa.2022.118619

[12] F. Fanian, M. Kuchaki Rafsanjani, M. Shokouhifar, *Combined fuzzy-metaheuristic framework for bridge health monitoring using UAV-enabled rechargeable wireless sensor networks*, Applied Soft Computing, **167** (2024), 112429. https://doi.org/10.1016/j.asoc.2024.112429

[13] B. R. HamaKarim, R. P. Mohammadiani, A. Sheikhahmadi, B. R. Hamakarim, M. Bahrami, *A method based on k-shell decomposition to identify influential nodes in complex networks*, The Journal of Supercomputing, **79**(14) (2023), 15597-15622. https://doi.org/10.1007/s11227-023-05296-y

[14] M. Ishaque, M. G. M. Johar, A. Khatibi, M. Yamin, *A novel hybrid technique using fuzzy logic, neural networks and genetic algorithm for intrusion detection system*, Measurement: Sensors, **30** (2023), 100933. https://doi.org/10.1016/j.measen.2023.100933

[15] W. Jia, M. Sun, J. Lian, S. Hou, *Feature dimensionality reduction: A review*, Complex and Intelligent Systems, **8**(3) (2022), 2663-2693. https://doi.org/10.1007/s40747-021-00637-x

[16] M. Joodaki, M. B. Dowlatshahi, N. Z. Joodaki, *An ensemble feature selection algorithm based on PageRank centrality and fuzzy logic*, Knowledge-Based Systems, **233** (2021), 107538. https://doi.org/10.1016/j.knosys.2021.107538

[17] T. R. Khalifa, A. M. El-Nagar, M. A. El-Brawany, E. A. G. El-Araby, M. El-Bardini, *A novel fuzzy Wiener-based nonlinear modelling for engineering applications*, ISA Transactions, **97** (2020), 130-142. https://doi.org/10.1016/j.isatra.2019.07.017

[18] D. Kreuzer, D. Beaini, W. L. Hamilton, V. Létourneau, P. Tossou, *Rethinking graph transformers with spectral attention*, arXiv, (2021). https://doi.org/10.48550/ARXIV.2106.03893

[19] Krishankumar R, R. Ks, A. B. Saeid, *A new extension to PROMETHEE under intuitionistic fuzzy environment for solving supplier selection problem with linguistic preferences*, Applied Soft Computing, **60** (2017), 564-576. https://doi.org/10.1016/j.asoc.2017.07.028

[20] D. Krleza, K. Fertalj, *Graph matching using hierarchical fuzzy graph neural networks*, IEEE Transactions on Fuzzy Systems, **25**(4) (2017), 892-904. https://doi.org/10.1109/TFUZZ.2016.2586962

[21] W. Li, H. Fujita, C. Zhang, S. F. Su, *Editorial: Fuzzy big data-driven computational intelligence models and applications*, International Journal of Fuzzy Systems, (2024), s40815-024-01821–0. https://doi.org/10.1007/s40815-024-01821-0

[22] F. Li, T. Zheng, Q. Cao, *Modeling and identification for practical nonlinear process using neural fuzzy network-based Hammerstein system*, Transactions of the Institute of Measurement and Control, **45**(11) (2023), 2091-2102. https://doi.org/10.1177/01423312221143777

[23] T. Lin, Y. Wang, X. Liu, X. Qiu, *A survey of transformers*, AI Open, **3** (2022), 111-132. https://doi.org/10.1016/j.aiopen.2022.10.001

[24] R. Liu, S. Duan, L. Xu, L. Liu, J. Li, Y. Zou, *A fuzzy transformer fusion network (FuzzyTransNet) for medical image segmentation: The case of rectal polyps and skin lesions*, Applied Sciences, **13**(16) (2023), 9121. https://doi.org/10.3390/app13169121

[25] Y. Liu, M. Liu, X. Xu, *Adaptive control design for fixed-time synchronization of fuzzy stochastic cellular neural networks with discrete and distributed delay*, Iranian Journal of Fuzzy Systems, **18**(6) (2021), 13-28. https://doi.org/10.22111/ijfs.2021.6330

[26] J. Lu, G. Ma, G. Zhang, *Fuzzy machine learning: A comprehensive framework and systematic review*, IEEE Transactions on Fuzzy Systems, **32**(7) (2024), 3861-3878. https://doi.org/10.1109/TFUZZ.2024.3387429

[27] G. Mali, S. Misra, *CORA: Cooperative communication and optimal resource allocation in multihop wireless multimedia sensor networks*, IEEE Internet of Things Journal, **11**(3) (2024), 4076-4084. https://doi.org/10.1109/JIOT.2023.3300774

[28] V. Ojha, A. Abraham, V. Snášel, *Heuristic design of fuzzy inference systems: A review of three decades of research*, Engineering Applications of Artificial Intelligence, **85** (2019), 845-864. https://doi.org/10.1016/j.engappai.2019.08.010

[29] K. Oono, T. Suzuki, *Graph neural networks exponentially lose expressive power for node classification*, Published as a Conference Paper at ICLR, (2020). https://doi.org/10.48550/ARXIV.1905.10947

[30] P. Pham, *An integrated simplicial neural network with neuro-fuzzy network for graph embedding*, International Journal of Machine Learning and Cybernetics, **16** (2025), 233-251. https://doi.org/10.1007/s13042-024-02201-8

[31] M. Rehyani Hamedani, S. W. Kim, *AdaSim: A recursive similarity measure in graphs*, in Proceedings of the 30th ACM International Conference on Information and Knowledge Management, Virtual Event Queensland Australia: ACM, (2021), 1528-1537. https://doi.org/10.1145/3459637.3482316

[32] M. Riyahi, A. Borumand Saeid, M. Kuchaki Rafsanjani, *Improved q-rung orthopair and T-spherical fuzzy sets*, Iranian Journal of Fuzzy Systems, **19**(3) (2022), 155-170. https://doi.org/10.22111/ijfs.2022.6949

[33] A. Saeedi, M. Kuchaki Rafsanjani, S. Yazdani, *Energy efficient clustering in IoT-based wireless sensor networks using binary whale optimization algorithm and fuzzy inference system*, The Journal of Supercomputing, **81**(1) (2025), 209. https://doi.org/10.1007/s11227-024-06556-1

[34] S. Salmani Pour Avval, N. D. Eskue, R. M. Groves, V. Yaghoubi, *Systematic review on neural architecture search*, Artificial Intelligence Review, **58**(3) (2025), 73. https://doi.org/10.1007/s10462-024-11058-w

[35] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders, K. R. Muller, *Explaining deep neural networks and beyond: A review of methods and applications*, Proceedings of the IEEE, **109**(3) (2021), 247-278. https://doi.org/10.1109/JPROC.2021.3060483

[36] P. Shaw, J. Uszkoreit, A. Vaswani, *Self-attention with relative position representations*, arXiv, (2018). https://doi.org/10.48550/ARXIV.1803.02155

[37] V. Snášel, M. Štěpnička, V. Ojha, P. N. Suganthan, R. Gao, L. Kong, *Large-scale data classification based on the integrated fusion of fuzzy learning and graph neural network*, Information Fusion, **102** (2024), 102067. https://doi.org/10.1016/j.inffus.2023.102067

[38] L. Sun, J. Hou, C. Xing, Z. Fang, *A robust Hammerstein-Wiener model identification method for highly nonlinear systems*, Processes, **10**(2) (2022), 2664. https://doi.org/10.3390/pr10122664

[39] T. Takagi, M. Sugeno, *Fuzzy identification of systems and its applications to modeling and control*, in Readings in Fuzzy Sets for Intelligent Systems, Elsevier, (1993), 387-403. https://doi.org/10.1016/B978-1-4832-1450-4.50045-6

[40] N. Talpur, S. J. Abdulkadir, H. Alhussian, M. H. Hasan, N. Aziz, A. Bamhdi, *Deep neuro-fuzzy system application trends, challenges, and future perspectives: A systematic survey*, Artificial Intelligence Review, **56**(2) (2023), 865-913. https://doi.org/10.1007/s10462-022-10188-3

[41] H. H. Tang, N. S. Ahmad, *Fuzzy logic approach for controlling uncertain and nonlinear systems: A comprehensive review of applications and advances*, Systems Science and Control Engineering, **12**(1) (2024), 2394429.https://doi.org/10.1080/21642583.2024.2394429

[42] J. Tavoosi, A. Mohammadzadeh, K. Jermsittiparsert, *A review on type-2 fuzzy neural networks for system identification*, Soft Computing, **25**(10) (2021), 7197-7212. https://doi.org/10.1007/s00500-021-05686-5

[43] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lió, Y. Bengio, *Graph attention networks*, arXiv, (2017). https://doi.org/10.48550/ARXIV.1710.10903

[44] T. Vo, *An integrated fuzzy neural supervision and attention-based graph neural network for improving network clustering*, Neural Computing and Applications, **35**(33) (2023), 24015-24035. https://doi.org/10.1007/s00521-023-08974-3

[45] X. Wei, G. Wang, B. Schmalz, D. F. T. Hagan, Z. Duan, *Evaluation of transformer model and self-attention mechanism in the Yangtze river basin runoff prediction*, Journal of Hydrology: Regional Studies, **47** (2023), 101438. https://doi.org/10.1016/j.ejrh.2023.101438

[46] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P. S. Yu, *A comprehensive survey on graph neural networks*, IEEE Transactions on Neural Networks and Learning Systems, **32**(1) (2021), 4-24. https://doi.org/10.1109/TNNLS.2020.2978386

[47] L. Xiao, X. Wu, G. Wang, *Social network analysis based on graph SAGE*, in 2019 12th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China: IEEE, (2019), 196-199. https://doi.org/10.1109/ISCID.2019.10128

[48] S. Xu, N. Feng, K. Liu, Y. Liang, X. Liu, *A weighted fuzzy process neural network model and its application in mixed-process signal classification*, Expert Systems with Applications, **172** (2021), 114642. https://doi.org/10.1016/j.eswa.2021.114642

[49] Y. Xue, et al., *Neural architecture search with progressive evaluation and sub-population preservation*, IEEE Transactions on Evolutionary Computation, (2024), 1-1. https://doi.org/10.1109/TEVC.2024.3393304

[50] Y. Xue, X. Han, F. Neri, J. Qin, D. Pelusi, *A gradient-guided evolutionary neural architecture search*, IEEE Transactions on Neural Networks and Learning Systems, **36**(3) (2025), 4345-4357. https://doi.org/10.1109/TNNLS.2024.3371432

[51] G. Yuan, B. Xue, M. Zhang, *An evolutionary neural architecture search method based on performance prediction and weight inheritance*, Information Sciences, **667** (2024), 120466. https://doi.org/10.1016/j.ins.2024.120466

[52] S. Yun, et al., *Graph transformer networks: Learning meta-path graphs to improve GNNs*, Neural Network, **153** (2022), 104-119. https://doi.org/10.1016/j.neunet.2022.05.026

[53] S. Yun, M. Jeong, R. Kim, J. Kang, H. J. Kim, *Graph transformer networks*, arXiv, (2019). https://doi.org/10.48550/ARXIV.1911.06455

[54] Y. Zhang, et al., *Takagi-Sugeno-Kang fuzzy system fusion: A survey at hierarchical, wide and stacked levels*, Information Fusion, **101** (2024), 101977. https://doi.org/10.1016/j.inffus.2023.101977

[55] S. Zhang, H. Tong, J. Xu, R. Maciejewski, *Graph convolutional networks: A comprehensive review*, Computational Social Networks, **6**(1) (2019), 11. https://doi.org/10.1186/s40649-019-0069-y

[56] J. Zhang, H. Zhang, C. Xia, L. Sun, *Graph-Bert: Only attention is needed for learning graph representations*, arXiv, (2020). https://doi.org/10.48550/ARXIV.2001.05140