The Journal of Systems and Software 216 (2024) 112139

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

On the effectiveness of hybrid pooling in mixup-based graph learning for language processing☆

Zeming Dong [a], Qiang Hu [b,*], Zhenya Zhang [a], Yuejun Guo [c], Maxime Cordy [b], Mike Papadakis [b], Yves Le Traon [b], Jianjun Zhao [a]

[a] Kyushu University, Japan
[b] University of Luxembourg, Luxembourg
[c] Luxembourg Institute of Science and Technology, Luxembourg

ARTICLE INFO

Dataset link: https://github.com/zemingd/HybridPool4Mixup

Keywords:
Hybrid pooling
Data augmentation
Graph learning
Manifold-mixup
Language processing

ABSTRACT

Graph neural network (GNN)-based graph learning has been popular in natural language and programming language processing, particularly in text and source code classification. Typically, GNNs are constructed by incorporating alternating layers which learn transformations of graph node features, along with graph pooling layers that use graph pooling operators (e.g., Max-pooling) to effectively reduce the number of nodes while preserving the semantic information of the graph. Recently, to enhance GNNs in graph learning tasks, *Manifold-Mixup*, a data augmentation technique that produces synthetic graph data by linearly mixing a pair of graph data and their labels, has been widely adopted. However, the performance of *Manifold-Mixup* can be highly affected by graph pooling operators, and there have not been many studies that are dedicated to uncovering such affection. To bridge this gap, we take an early step to explore how graph pooling operators affect the performance of Mixup-based graph learning. To that end, we conduct a comprehensive empirical study by applying *Manifold-Mixup* to a formal characterization of graph pooling based on 11 graph pooling operations (9 hybrid pooling operators, 2 non-hybrid pooling operators). The experimental results on both natural language datasets (Gossipcop, Politifact) and programming language datasets (JAVA250, Python800) demonstrate that hybrid pooling operators are more effective for *Manifold-Mixup* than the standard Max-pooling and the state-of-the-art graph multiset transformer (GMT) pooling, in terms of producing more accurate and robust GNN models.

Editor's note: Open Science material was validated by the Journal of Systems and Software Open Science Board.

1. Introduction

Since texts, as well as source code, can be represented as graph-structured data (Huang et al., 2019; Allamanis et al., 2018a), *graph neural network (GNN)*-based *graph learning* has been increasingly applied for both *natural language processing (NLP)* (Wu et al., 2021), and *programming language (PL)* understanding (Dinella et al., 2020; Wang et al., 2020a). The application of GNNs has achieved remarkable results, e.g., Allamanis et al. (2018b) utilize GNNs to learn the syntax tree and data flow representations of source code, by which they manage to accomplish several software engineering tasks, such as code completion and defect detection.

Typically, high-quality training data, including features and their corresponding labels, are necessary to train GNN models with competitive performance. However, preparing the labeled data is often not easy, especially in the context of source code labeling that requires advanced expertise (Zhou et al., 2019). To alleviate the data labeling issue, *data augmentation* has been proposed to enhance training data by modifying original data. As the state-of-the-art in data augmentation, *Mixup* (Zhang et al., 2018a) achieves impressive results in different tasks. Take image classification for an example: *Mixup* synthesizes new images and labels as additional training data by first selecting two raw images at random from the original training data and then linearly mixing their features and labels. Recent research (Wang et al., 2021; Dong et al., 2023a) demonstrates that *Manifold-Mixup*, the specialized application of *Mixup* on graph-structured data (Dong et al., 2023a), focusing on interpolating graph-level embeddings, can also achieve great performance for graph-structured data. With the success of *Manifold-Mixup*, utilizing Mixup-based data augmentation in graph learning has emerged as a mainstream paradigm.

☆ Editor: Nicole Novielli.

* Corresponding author.
E-mail addresses: dongzm9312@gmail.com (Z. Dong), qianghu0515@gmail.com (Q. Hu).

https://doi.org/10.1016/j.jss.2024.112139
Received 12 June 2023; Received in revised form 11 March 2024; Accepted 14 June 2024
Available online 17 June 2024

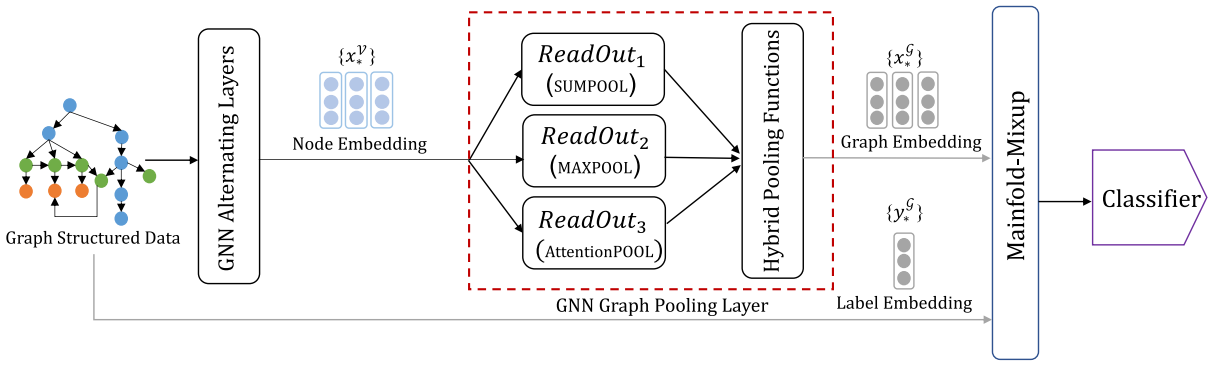


Fig. 1. Overview of Mixup-based graph learning via hybrid pooling.

Table 1
Nine types of hybrid pooling.

	Hybrid pooling layer		
	Pooling function	Pooling operator	Description
Type1	$\mathcal{M}_{sum}$	$\mathcal{P}_{att}, \mathcal{P}_{max}$	Perform matrix addition on the vectors generated by $\mathcal{P}_{att}$ and $\mathcal{P}_{max}$
Type2	$\mathcal{M}_{mul}$	$\mathcal{P}_{att}, \mathcal{P}_{max}$	Perform Hadamard product on the vectors generated by $\mathcal{P}_{att}$ and $\mathcal{P}_{max}$
Type3	$\mathcal{M}_{concat}$	$\mathcal{P}_{att}, \mathcal{P}_{max}$	Concatenate the vectors that are generated by $\mathcal{P}_{att}$ and $\mathcal{P}_{max}$
Type4	$\mathcal{M}_{sum}$	$\mathcal{P}_{att}, \mathcal{P}_{sum}$	Perform matrix addition on the vectors generated by $\mathcal{P}_{att}$ and $\mathcal{P}_{sum}$
Type5	$\mathcal{M}_{mul}$	$\mathcal{P}_{att}, \mathcal{P}_{sum}$	Perform Hadamard product on the vectors generated by $\mathcal{P}_{att}$ and $\mathcal{P}_{sum}$
Type6	$\mathcal{M}_{concat}$	$\mathcal{P}_{att}, \mathcal{P}_{sum}$	Concatenate the vectors that are generated by $\mathcal{P}_{att}$ and $\mathcal{P}_{sum}$
Type7	$\mathcal{M}_{sum}$	$\mathcal{P}_{sum}, \mathcal{P}_{max}$	Perform matrix addition on the vectors generated by $\mathcal{P}_{sum}$ and $\mathcal{P}_{max}$
Type8	$\mathcal{M}_{mul}$	$\mathcal{P}_{sum}, \mathcal{P}_{max}$	Perform Hadamard product on the vectors generated by $\mathcal{P}_{sum}$ and $\mathcal{P}_{max}$
Type9	$\mathcal{M}_{concat}$	$\mathcal{P}_{sum}, \mathcal{P}_{max}$	Concatenate the vectors that are generated by $\mathcal{P}_{sum}$ and $\mathcal{P}_{max}$

As indicated by existing studies (Dong et al., 2023a; Verma et al., 2019), Mixup-based graph learning is mainly influenced by two factors, namely, (1) the hyperparameters in *Mixup* itself, such as the *Mixup ratio* that balances the proportion of the source data, and (2) the *Mixup* strategies that are associated with representation generation. Over these factors, the hyperparameter issue is a common one across several different Mixup-applied fields, and has been extensively studied in fields such as image classification (Zhang et al., 2022, 2021). However, the second issue about Mixup strategies is highly correlated to the context in which Mixup is applied, and for graph-structured data, the influence of such an issue has not been well studied.

In the context of Mixup-based graph learning, as shown in Fig. 1, *Manifold-Mixup* is fed with the inputs from the *graph pooling layer*, which use graph pooling operators (e.g., Max-pooling) to produce coarsened representations of the given graph while preserving its semantic information. Namely, this layer is the key to representation generation of *Manifold-Mixup*, in that *Manifold-Mixup* generates augmented training data by interpolating these representations. Therefore, the performance of *Manifold-Mixup* can be highly affected by the graph pooling operators. Recent works (Knyazev et al., 2019; Mesquita et al., 2020; Grattarola et al., 2022) have attempted to systematically analyze the importance of graph pooling in representation generation; however, the following question, namely, *how different graph pooling operators affect the effectiveness of Mixup-based graph learning*, still remains open.

In this paper, we tackle this problem by empirically analyzing the difference when *Mixup* is applied in different graph representations generated by different graph pooling operators. Specifically, we focus on two types of graph pooling methods, namely, standard pooling methods and a unifying formulation of hybrid (mixture) pooling operators. For the standard pooling, the Max-pooling, which is the most widely used one (Mesquita et al., 2020), and the state-of-the-art *graph multiset transformer pooling* (GMT) (Baek et al., 2021) which is a global pooling layer based on multi-head attention and capturing node interactions based on structural dependencies, are considered. For the hybrid pooling, we extend the prior work (Grattarola et al., 2022; Nguyen et al., 2022) and design 9 types of hybrid pooling strategies, and more details are introduced in Table 1. Here, GMT and hybrid pooling operators are considered more advanced strategies. We conduct empirical experiments to evaluate the effectiveness of graph learning using *Manifold-Mixup* (Verma et al., 2019), under different hybrid pooling operators. In total, our experiments cover diverse types of datasets, including two programming languages (JAVA and Python) and one natural language (English), and consider different tasks, two widely-studied graph-level classification tasks (program classification and Fake news detection), and six GNN model architectures. Based on that, we answer the following research questions:

RQ1: How effective are hybrid pooling operators for enhancing the accuracy of Mixup-based graph learning? The results on NLP datasets (Gossipcop and Politifact used for fake news detection) show that the hybrid pooling operator Type 1 ($\mathcal{M}_{sum}(\mathcal{P}_{att}, \mathcal{P}_{max})$) outperforms GMT by up to 4.38% accuracy. On PL datasets (JAVA250 and Python800 used for problem classification), also the hybrid pooling operator Type 1 ($\mathcal{M}_{sum}(\mathcal{P}_{att}, \mathcal{P}_{max})$) surpasses GMT by up to 2.36% accuracy.

RQ2: How effective are hybrid pooling operators for enhancing the robustness of Mixup-based graph learning? The results demonstrate that in terms of robustness, the hybrid pooling operator Type 6 ($\mathcal{M}_{concat}(\mathcal{P}_{att}, \mathcal{P}_{sum})$) surpasses GMT by up to 23.23% in fake news detection, while the hybrid pooling operator Type 1 ($\mathcal{M}_{sum}(\mathcal{P}_{att}, \mathcal{P}_{max})$) outperforms GMT by up to 10.23% in program classification.

RQ3: How does the hyperparameter setting affect the effectiveness of Manifold-Mixup when hybrid pooling operators are applied? According to Zhang et al. (2018a), the hyperparameter λ denotes the interpolation ratio, and it is sampled from a *Beta* distribution with a shape parameter α ($\lambda \sim Beta(\alpha, \alpha)$). Existing works (Dong et al., 2023a; Verma et al., 2019) show that the hyperparameter setting λ affects the performance of Mixup. Therefore, we study the effectiveness of *Manifold-Mixup* when using hybrid pooling operators under different hyperparameters of *Mixup*. Experimental results indicate that a smaller value of the hyperparameter leads to better robustness and accuracy.

In summary, the contributions of this paper are as follows:

- This is the first work that explores the potential influence of graph pooling operators on Mixup-based graph-structured data

augmentation. To facilitate reproducibility, our code and data are available online.[1]
- We discuss and further extend the hybrid pooling operators from existing works.
- The comprehensive empirical analysis demonstrates that hybrid pooling is a better way for Mixup-based graph-structured data augmentation.

2. Background and related work

2.1. Graph data classification

Researchers have proposed multiple approaches for the text classification task that analyze the data based on its graph structure. In which Yao et al. (2019) constructed text graph data by using the words and documents as nodes. To further enhance text classification performance, Zhang et al. (2020a) proposed the graph-based word interaction to capture the contextual word relationships. Similar to text data, by capturing the relationship of different components (e.g., variables and operators) in the code, source code data can also be represented structurally as the graph data (Wang et al., 2020a; Dong et al., 2023b). Zhou et al. (2019) mainly integrated four separate subgraph representations of source code into one joint graph data. Furthermore, to advance the generalization, Allamanis et al. (2021) offered four code rewrite rules, such as variable renaming, comment deletion, etc., as a data augmentation for graph-level program classification.

Different from the above works, our study focuses on enhancing the performance of graph classification with Mixup-based data augmentation.

2.2. Graph pooling operators

Graph pooling (Ying et al., 2018; Grattarola et al., 2022) plays a crucial role in capturing relevant structure information of the entire graph. Existing works (Atwood and Towsley, 2016; Simonovsky and Komodakis, 2017; Xu et al., 2019) have proposed the basic graph pooling methods, such as summing or averaging all of the node features. However, such pooling methods treat every node information identically, which could lose the structural information. To solve this problem, researchers (Zhang et al., 2018b; Gao and Ji, 2019; Lee et al., 2019; Cangea et al., 2018) dropped nodes with lower scores using a learnable scoring function, which can compress the graph and alleviate the impact of irrelevant nodes to save the important structural information. Additionally, to locate the tightly related communities on a graph, recent works (Ma et al., 2019; Wang et al., 2019; Bianchi et al., 2020; Yuan and Ji, 2020) have considered the graph pooling as the node clustering problem, where nodes were specifically aggregated to the same cluster. To combine these advantages, Ranjan et al. (2020) first clustered nearby nodes locally and dropped clusters with lower scores. In addition, there existed a kind of attention-based pooling methods (Li et al., 2016, 2019; Bahdanau et al., 2014; Baek et al., 2021) that scored nodes with an attention mechanism to weight the relevance of nodes to the current graph-level task. Besides, different from the above standard pooling methods, Nguyen et al. (2022) leveraged a mixture of Sum-pooling and Max-pooling methods for graph classification.

In our study, we examine the application of *Manifold-Mixup* in graph-level classification, incorporating both hybrid pooling and standard pooling techniques. Moreover, different from existing research (Nguyen et al., 2022), we specifically include attention pooling because it has been proven effective for GNN model training (Lee et al., 2019; Li et al., 2016). We extended from the original three different types of hybrid pooling operators (Nguyen et al., 2022) to the current nine.

[1] https://github.com/zemingd/HybridPool4Mixup

2.3. Mixup

Due to its effectiveness in graph-structured data processing and the promising performance on graph-specific downstream tasks, e.g., graph classification, GNN has recently received considerable attention. Meanwhile, as a sophisticated data augmentation method, *Mixup* (Zhang et al., 2018a) was initially proposed and implemented within the domain of image classification (Shorten and Khoshgoftaar, 2019). Specifically, Mixup randomly selects a pair of images from the training data, linearly combines their features and labels, and synthesizes a new image and label, which are treated as augmented training data. By combining two different data points linearly, Mixup effectively smooths the data distribution in the feature space. This smoothing process helps mitigate the sharpness of decision boundaries between different classes, reducing susceptibility to overfitting (Dong et al., 2024). Due to its remarkable efficacy in classification tasks, it has subsequently been gradually applied to NLP (Guo et al., 2019) and source code learning (Dong et al., 2023a).

In the NLP, Guo et al. (2019) proposed two basic strategies of Mixup for augmenting data. One was wording embedding-based, and another was sentence embedding-based. After these two different kinds of embedding, the feature of input data can be mixed to synthesize the new data in vector space. To solve the difficulty in mixing text data in the raw format, Sun et al. (2020) mixed text data from transformer-based pre-trained architecture. Chen et al. (2020) increased the size of augmented samples by interpolating text data in hidden space. Zhang et al. (2020b) generated extra labeled sequences in each iteration to augment the scale of training data. Unlike previous work, some researchers considered the raw text itself to augment the input data. Yoon et al. (2021) synthesized the new text data from two raw input data by span-based mixing to replace the hidden vectors. In source code learning, Dong et al. (2023a) proposed a mixup-based data augmentation method that linearly interpolates the features of a pair of programs as well as their labels for the GNN model training.

In our work, we do not simply employ *Mixup* for graph-structured data classification. Instead, we explore how different graph pooling operators affect the effectiveness of *Mixup*.

2.4. Empirical study on data augmentation

Recently, there has been a surge of empirical studies exploring the topic of data augmentation. In the NLP, Konno et al. (2020) presented an empirical analysis to evaluate the effectiveness of contextual data augmentation (CDA) in improving the quality of the augmented training data, in comparison to *[MASK]-based augmentation* and *linguistically-controlled masking*. To assist practitioners in selecting suitable augmentation strategies, Chen et al. (2023) conducted a comprehensive empirical study on 11 datasets, encompassing topics such as news classification, inference tasks, paraphrasing tasks, and single-sentence tasks. In source code learning, Yu et al. (2022) conducted an empirical study on three program-related downstream tasks, namely method naming, code commenting, and clone detection. The study aimed to validate the effectiveness of data augmentation that is designed by 18 program transformation methods that preserve both semantics and syntax-naturalness. Dong et al. (2023b) presented a meticulous empirical study that aimed to evaluate the effectiveness of data augmentation methods that were adapted from the domains of NLP and graph learning for source code learning. The study rigorously examined the impact of these augmentation techniques on enhancing the performance of various tasks related to source code analysis and understanding.

Different from existing empirical studies, our work focuses primarily on examining the influence of graph pooling operators on Mixup-based data augmentation that has received limited attention thus far.

3. Mixup-based graph learning via hybrid pooling

3.1. Overview

Fig. 1 provides an overview of GNNs for graph classification, demonstrating the application of *Manifold-Mixup* after the hybrid pooling layer. Concretely, first, GNNs process the input graph-structured data and transform them into node attributes $\{x_i^{\mathcal{V}}\}_{i=1}^{n}$, where $\mathcal{V}$ is the vertex set. Then, after learning the features of each node, the hybrid pooling layer produces the entire graph embedding by utilizing three fundamental types of readout functions: Sum-pooling (SUMPOOL), Max-pooling (MAXPOOL), and Attention-pooling (AttentionPOOL). Finally, as shown in Eq. (1), *Manifold-Mixup* is applied to randomly mix two selected graph embeddings $\widetilde{x_i^{\mathcal{G}}}$, $\widetilde{x_j^{\mathcal{G}}}$ and their ground truth labels $y_i^{\mathcal{G}}$, $y_j^{\mathcal{G}}$ with one-hot values as the new training set. This augmented training set is then used for training the classifier. Especially, the utilization of *Manifold-Mixup* employing the hybrid pooling layer is depicted as follows:

$$\begin{aligned} \widetilde{x_{mix}^{\mathcal{G}}} &= \lambda \widetilde{x_i^{\mathcal{G}}} + (1-\lambda)\widetilde{x_j^{\mathcal{G}}} \\ y_{mix}^{\mathcal{G}} &= \lambda y_i^{\mathcal{G}} + (1-\lambda) y_j^{\mathcal{G}} \end{aligned} \tag{1}$$

where, $\widetilde{x^{\mathcal{G}}}$ is the graph embedding and $y^{\mathcal{G}}$ refers to its label embedding with one-hot value, wherein λ means the *Mixup* ratio. According to Zhang et al. (2018a), Guo et al. (2019), λ is sampled from a Beta distribution with a shape parameter α ($\lambda \sim Beta(\alpha, \alpha)$).

3.2. Graph-level hybrid pooling

In recent research (Wang et al., 2021), several methods have been proposed to represent graphs at different levels of granularity. These include node-level representation (e.g., node embeddings) and graph-level representation (e.g., graph embeddings), each serving different purposes: node embeddings can be utilized for tasks such as node classification and link prediction, while graph embeddings can be used for graph classification and graph generation (Wu et al., 2021). Our work specifically focuses on graph-level representation and its related downstream tasks, e.g., graph classification.

Generally, GNNs learn the graph representation by exploiting the graph structure as an inductive bias, and the GNN architecture proposed by Gilmer et al. (2017) is the most popular and widely used in the applications. Briefly speaking, first, given a node with its initialized information, GNN computes the representation by iteratively aggregating its adjacent nodes (**Aggregate**). Then, it combines this aggregated representation with the existing node representation (**Update**). After that, a final representation of the complete graph is created by pooling learned features of nodes (**Pooling**). The primary areas where the various models differ are how they handle aggregation, update, and pooling.

Formally, given a graph-structured data $\mathcal{G}(\mathcal{V}, \varepsilon)$, where $\mathcal{V}$ is the vertex set, and ε represents the edge set, we simply formulate the entire graph representation $x^{\mathcal{G}}$ as follows:

$$\begin{aligned} \mathbf{H}_u^k &= \mathbf{GNN}\left(\mathbf{H}_u^{k-1}, W, \sum_{v \in \mathcal{N}(u)} \mathbf{H}_v^{k-1}\right) \\ \mathbf{H}^{\mathcal{G}} &= \mathcal{P}(H, \mathcal{V}) \quad \text{s.t.} \quad H = \mathbf{H}_u^k \quad (u \in \mathcal{V}) \end{aligned} \tag{2}$$

where $\mathbf{H}_u^{k-1}$ and $\mathbf{H}_v^{k-1}$ denote the matrix representation of node u and v ($u, v \in \mathcal{V}$, $uv \in \varepsilon$) at the $(k-1)$th iteration, W represents the trainable parameter matrix, $\mathcal{N}(u)$ is the set of neighbor nodes of u, and $\mathcal{P}$ is a graph pooling operator that produces $\mathbf{H}^{\mathcal{G}}$, the latent vector of the entire graph $x^{\mathcal{G}}$. In Eq. (2), firstly, neural network $\mathbf{GNN}(\cdot)$ is used to iteratively update the latent vector of each node via the message aggregated from the neighborhood $\sum_{v \in \mathcal{V}(u)} \mathbf{H}_v^{k-1}$. After that, the pooling operator $\mathcal{P}(\cdot)$ is used to construct the vector representation of the entire graph, which captures the global information, after k steps of iteration.

In our study, we consider three standard pooling operators $\mathcal{P}$, which are as follows:

$$\begin{aligned} \mathcal{P}_{att}(H, \mathcal{V}) &= \sum_{u \in \mathcal{V}} \sigma\left(W\mathbf{H}_u^k + b\right) \odot \phi\left(W\mathbf{H}_u^k + b\right) \\ \mathcal{P}_{sum}(H, \mathcal{V}) &= \sum_{u \in \mathcal{V}} \mathbf{H}_u^k \\ \mathcal{P}_{max}(H, \mathcal{V}) &= \max_{u \in \mathcal{V}} \mathbf{H}_u^k \end{aligned} \tag{3}$$

where ϕ represents the nonlinear activation function, $\odot$ means the Hadamard Product and $\mathbf{H}_i^k$ refers to the final vector representation of the ith node. Here, $\sigma\left(W\mathbf{H}_i^k + b\right)$ acts as a soft attention mechanism. We leverage the global attention pooling $\mathcal{P}_{att}$ as it better captures relevant global features for graph-level tasks (Li et al., 2016).

Finally, we examine three hybrid pooling functions $\mathcal{M}_{sum}$, $\mathcal{M}_{mul}$ and $\mathcal{M}_{concat}$, defined as follows:

$$\begin{aligned} \mathcal{M}_{sum}\left(\mathcal{P}_i, \mathcal{P}_j\right) &= \mathbf{H}_{\mathcal{P}_i}^{\mathcal{G}} + \mathbf{H}_{\mathcal{P}_j}^{\mathcal{G}} \\ \mathcal{M}_{mul}\left(\mathcal{P}_i, \mathcal{P}_j\right) &= \mathbf{H}_{\mathcal{P}_i}^{\mathcal{G}} \odot \mathbf{H}_{\mathcal{P}_j}^{\mathcal{G}} \\ \mathcal{M}_{concat}\left(\mathcal{P}_i, \mathcal{P}_j\right) &= T\left(\left[\mathbf{H}_{\mathcal{P}_i}^{\mathcal{G}} \parallel \mathbf{H}_{\mathcal{P}_j}^{\mathcal{G}}\right]\right) + b \end{aligned} \tag{4}$$

where $\mathbf{H}_{\mathcal{P}_i}^{\mathcal{G}}$, $\mathbf{H}_{\mathcal{P}_j}^{\mathcal{G}}$ means the embedding vectors of the entire graph, respectively produced by two different pooling operators $\mathcal{P}_i, \mathcal{P}_j$ ($\mathcal{P}_i \neq \mathcal{P}_j$), as defined in Eq. (3). Here, $\parallel$ denotes the vector concatenation operation, and T is a linear transformation matrix to reduce the dimension $\left(\mathbb{R}^{2d} \rightarrow \mathbb{R}^{d}\right)$.

To conclude, the hybrid pooling layer, which consists of the pooling function and pooling operator, can be expressed as follows:

$$\widetilde{H^{\mathcal{G}}} = \sum_{u \in \mathcal{V}} \mathcal{M}_n(\mathcal{P}, H_u^k) \tag{5}$$

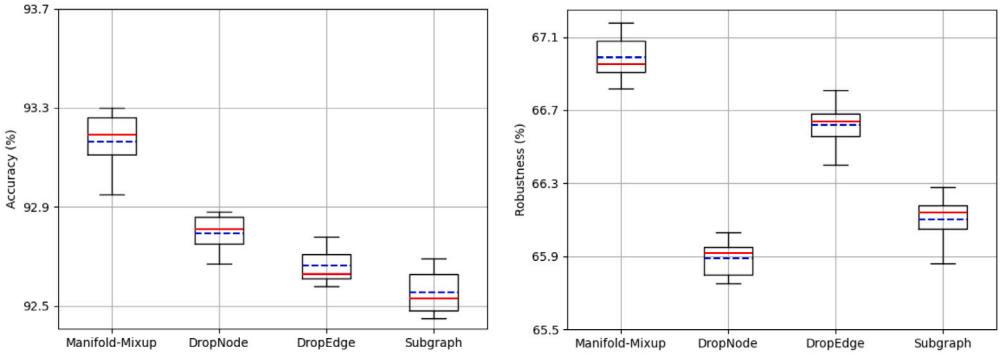
where $\mathcal{M}_n \in \{\mathcal{M}_{sum}, \mathcal{M}_{mul}, \mathcal{M}_{concat}\}$ and $\widetilde{H^{\mathcal{G}}}$ is the matrix representation of the graph data $\mathcal{G}$ produced by the hybrid pooling layer. To more clearly illustrate the design, Table 1 lists the 9 different types of hybrid pooling operators, which are considered in our study.

4. Experimental setup

4.1. Case study

We conduct a case study to evaluate the effectiveness of existing graph data augmentation methods, in terms of accuracy and robustness. Specifically, (1) we perform a survey to collect graph data augmentation methods from existing studies (Zhao et al., 2023; Dong et al., 2024). As a result, graph data augmentation methods including *Manifold-Mixup* (Verma et al., 2019), *DropNode* (Feng et al., 2020), *DropEdge* (Rong et al., 2020), and *Subgraph* (Wang et al., 2020b), which are adopted in our experiment. (2) We subsequently apply these methods to train GCN using the NLP dataset Gossipcop-Profile. In particular, the model training is repeated five times to mitigate the influence of randomness. (3) Finally, we record the results of each model training. Fig. 2 represents our experimental results. From the left part of Fig. 2, we can see that Manifold-Mixup shows the best performance compared to all considered graph data augmentation methods in the evaluation of accuracy. Moving to robustness, we observe a similar finding where Manifold-Mixup can consistently produce a more robust GCN model compared to other methods. Interestingly, although Subgraph showing lower accuracy improvement performance compared to DropNode, it exhibits better robustness improvement.

Manifold-Mixup has demonstrated its effectiveness on both graph data (Wang et al., 2021; Zhao et al., 2023) and PL data (Dong et al., 2024). The findings from our experiments are consistent with those of existing studies. By interpolating two different data points in the hidden space, Manifold-Mixup can help the GNN model learn more about the underlying data structure, thereby enhancing the accuracy performance. Additionally, the features generated by Manifold-Mixup result in more evenly distributed class-specific representations compared to



(a) Effectiveness of Accuracy (b) Effectiveness of Robustness

Fig. 2. A case study of graph data augmentation methods (Model: GCN, Dataset: Gossipcop-Profile).

other data augmentation methods such as Subgraph and DropNode. The smoothing effect on the decision boundary of classification induced by mixup can reduce the impact of these noises on the model's behavior, thereby generating more robust GNN models (Verma et al., 2019; Dong et al., 2024).

As mentioned in Section 1, the performance of Manifold-Mixup is mainly influenced by two factors including the Mixup ratio and Mixup strategies that are associated with the graph representation. While the former has been extensively studied, the impact of graph representations on Manifold-Mixup is rarely discussed. Therefore, considering its superior performance, we select Manifold-Mixup from existing graph data augmentation in our study and explore the potential of graph representations to further improve the performance of Manifold-Mixup.

Finding: Manifold-Mixup shows the best performance compared to other existing graph data augmentation methods including Subgraph, DropNode, and DropEdge in terms of accuracy and robustness improvement.

4.2. Study design

As discussed in Section 1, the success of Mixup, especially Manifold-Mixup, has led to the widespread adoption of Mixup-based data augmentation in graph learning, establishing it as a mainstream paradigm (Zhao et al., 2023). Therefore, in our study, our main focus is on exploring the potential of hybrid pooling operators for Mixup-based graph learning to enhance both accuracy and robustness performance. To assess the effectiveness of hybrid pooling operators, as introduced in Section 3.2, in Mixup-based graph learning, we design three research questions, as follows:

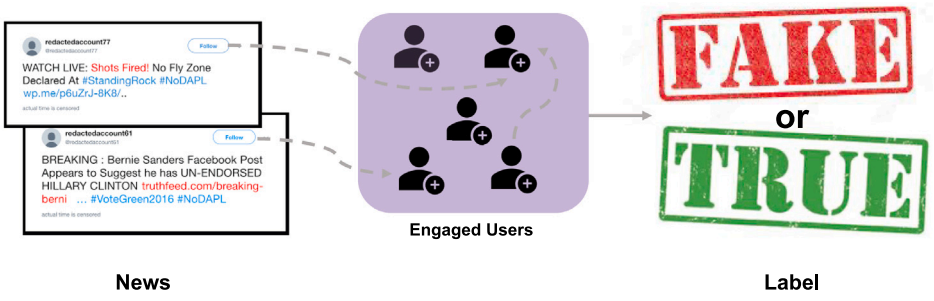
- **RQ1: How effective are hybrid pooling operators for enhancing the accuracy of Mixup-based graph learning?** We first explore whether hybrid pooling operators are able to help graph learning with *Manifold-Mixup* improve the accuracy of GNN models or not. Accuracy which calculates the percentage of correctly identified data among the total number of data in the given test data is the basic metric for evaluating the performance of trained models. Therefore, in this RQ we first evaluate the accuracy performance of hybrid pooling operators (as introduced in Section 3.2) on the NLP binary classification dataset and PL multi-classification dataset, compared to baseline approaches including Max-pooling and the state-of-the-art GMT pooling operators (as introduced in 2.2).
- **RQ2: How effective are hybrid pooling operators for enhancing the robustness of Mixup-based graph learning?** Robustness is another important metric to reflect the generalization ability of the trained GNN models in practice (Xu and Mannor, 2012; Tu, 2000; Neyshabur et al., 2017). In this RQ, we first generate robust test sets by the method (Papp et al., 2021) that randomly drops the edge-connectivity (i.e., the local connections between a node and its neighboring nodes) of each graph data from the original sets and then calculate the percentage of correctly classified data from this new robust set, to evaluate the robustness of hybrid pooling operators when applied to Mixup-based graph learning (Fabian et al., 2015; Oehlers and Fabian, 2021).
- **RQ3: How does the hyperparameter setting affect the effectiveness of Manifold-Mixup when hybrid pooling operators are applied?** The hyperparameter λ (as introduced in Section 3.1) affects the performance including both accuracy and robustness of Mixup-based graph learning (Zhang et al., 2018a; Dong et al., 2023a; Verma et al., 2019). Therefore, in this RQ we investigate whether this conclusion also holds for Mixup-based graph learning when hybrid pooling operators are applied. Concretely, we evaluate the accuracy and robustness of nine hybrid pooling operators under the setting of the *Mixup* ratio α from 0.05 to 0.5 in 0.05 intervals.

4.3. Task, dataset, and model

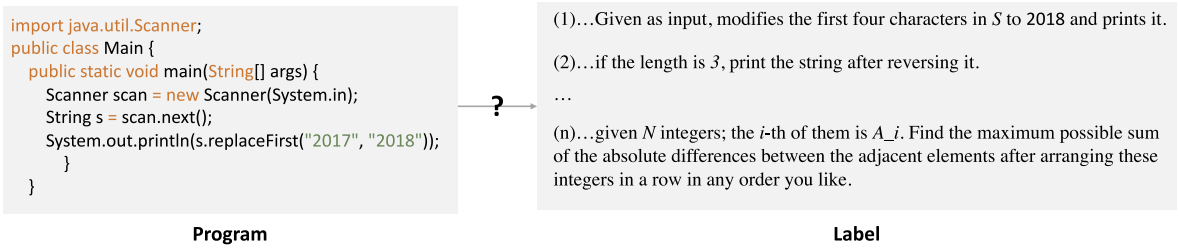
Task. In our study, we focus on two widely studied tasks including fake news detection and program classification. We provide a simple example to understand these two tasks, as depicted in Fig. 3. The proliferation of fake news on the internet presents a pressing social concern. To address this issue, researchers generally conceptualize Twitter users and news articles as nodes within a social network graph, with (re)tweeting actions represented as edges. Positioned at the intersection of graph learning and NLP, GNNs are applied to propagate text embeddings of nodes and aggregate them for binary classification (i.e., Fake or True) (Dou et al., 2021). Since the problem classification task is relatively new in the PL field, we also use one example to better explain it. When provided with a series of problems accompanied by detailed descriptions and their corresponding candidate source code, trained models can effectively identify the problem that the program solves. Moreover, representing the program as a graph allows for capturing semantic content (Allamanis et al., 2018a; Dong et al., 2024). Therefore, GNNs can be applied to learn the graph information of programs. Program classification task involves servers that automatically estimate the functionality of programs, which is crucial for software reuse (Puri et al., 2021).

Dataset. We conduct our study on both traditional NLP and PL tasks. For NLP, we consider User Preference-aware Fake News Detection (UPFD) (Dou et al., 2021), which is a text-level binary classification problem. For PL, we consider function-level problem multi-classification task provided by Project_CodeNet (Puri et al., 2021). Two popular programming languages, Java and Python, are included in our study.

UPFD contains two sets of tree-structured fake and real news propagation graphs that are derived from Twitter. Given a single graph,



(a) Fake news detection



(b) Program classification

Fig. 3. Examples of Fake news detection task and Program classification task.

the source news is represented by the root node, and the leaf nodes represent Twitter users who retweeted the root news. Edges reflect the (1) connection between users and the (2) connection between the user and root news that have been retweeted. Moreover, UPFD also includes 4 different node feature types. These are ❶ *Profile* feature, a 10-dimensional vector derived from ten user profile attributes; ❷ *Spacy* feature, a 300-dimensional vector encoded using spacy Word2Vec; ❸ *Bert* feature, a 768-dimensional vector encoded by BERT (Bidirectional Encoder Representations from Transformers); ❹ *Content* feature, a 310-dimensional vector obtained by summing the *Spacy* and *Profile* features. Fake news detection is commonly approached as a binary classification task, where GNNs provide a negative or positive prediction given a graph of user preferences. Project_CodeNet provides two datasets, JAVA250 and Python800. JAVA250 consists of 250 classification problems, and each problem has 300 Java programs. Python800 contains 800 classification problems, and each problem has 300 Python programs. In addition, we transformed the raw source code data from both datasets into a graph representation by utilizing a simplified parse tree. We follow the same process as the original paper to divide the datasets into training, validation, and test sets.

Model. We build 4 different GNN architectures for each dataset. In the text classification, we follow the recommendation of Dou et al. (2021) and build Graph Convolution Network (GCN) (Kipf and Welling, 2017), Graph Isomorphism Network (GIN) (Xu et al., 2019), Graph Attention Network (GAT) (Velickovic et al., 2017), and GraphSAGE (Hamilton et al., 2017) models. GCN is a specialized variant of a convolution neural network that is dedicated to operating graph-structured data. GAT mainly uses the attention mechanism for graph message passing. GIN is designed to generalize the Weisfeiler–Lehman (WL) test. GraphSAGE effectively generates node embeddings for previously undiscovered data by utilizing node feature information. In program classification, we use GNN models (GCN, GCN-Virtual, GIN, and GIN-Virtual) provided by Puri et al. (2021). Especially to enhance the aggregation phase of GNNs, virtual nodes, which involve adding an artificial node to each network and connecting it in both directions to all other graph nodes, are offered. To sum up, Table 2 shows the details of used datasets and models in our experiment.

4.4. Experiment settings

We implement the hybrid pooling layer based on the open-source library **PyG** (Pytorch Geometric) (Fey and Lenssen, 2019). The training epochs we set for Project_CodeNet and UPFD are 100 and 200, respectively. The batch size is set to 80 for Project_CodeNet and 128 for UPFD. For the optimizer, we use Adam (Kingma and Ba, 2014) with the learning rate 10^{-3} for all the models. For the Mixup ratio, $\alpha = 0.1$ is our default setting. To alleviate overfitting, we adopt early stopping with patience 20. To reduce the impact of randomness, following the convention in existing works (Dong et al., 2023a; Puri et al., 2021), we train each GNN model five times by using different random seeds. We present the average experimental results with standard deviation in the later Section. All the experiments were conducted on a server with 2 GPUs of NVIDIA RTX A6000.

5. Results analysis

We first study the influence of graph pooling operators on the GNN training with *Manifold-Mixup* in RQ1 and RQ2, then we go deeper into the hyperparameter settings of *Manifold-Mixup* in RQ3.

5.1. RQ1: How effective are hybrid pooling operators for enhancing the accuracy of mixup-based graph learning?

Accuracy analysis. The left part in Table 3 presents the detailed test accuracy results of the GCN model on NLP datasets. From the results, first, we can see that *Manifold-Mixup* is a powerful technique for augmenting graph-structured data. Compared to the model trained without *Manifold-Mixup* (*No Aug* in the tables), GCN model training with data augmentation is more accurate and robust. Then, concerning different pooling operators, we can see that the basic and widely used Max-pooling method (MAXPOOL) and the state-of-the-art GMT pooling operators (GMT) are not the best choices for GCN training with *Manifold-Mixup*. There are always some hybrid pooling operators that achieve better results than them. More specifically, 8 out of 9 hybrid pooling operators outperform MAXPOOL on the Gossipcop-Profile. Similarly, on the Politifact-Bert, 5 out of 9 hybrid pooling operators achieve higher performance than MAXPOOL. Among these results, the

Table 2
Details of datasets and DNNs. #Training, #Ori Test, and #Robust Test represent the number of training data, original test data, and test data for generalization evaluation, respectively.

Dataset	Task	#Training	#Clean test	#Robust test	Model
Politifact	Fake news detection	62	221	314	GCN GIN
Gossipcop	Fake news detection	1,092	3,826	5,464	GAT GraphSAGE
JAVA250	Problem classification	48,000	15,000	75,000	GCN GCN-Virtual
Python800	Problem classification	153,600	48,000	240,000	GIN GIN-Virtual

Table 3
Effectiveness of nine types of hybrid pooling w.r.t. test accuracy ↑ (average ± standard deviation, %) on original test data and robustness ↑ (average ± standard deviation, %) on robust test data of trained GCN models on NLP datasets. No Aug: without Manifold-Mixup data augmentation. A gray background highlights results that are better than MAXPOOL. The best average results are marked in red color.

		Test Accuracy					Robustness				
		Profile	Spacy	Bert	Content	Average	Profile	Spacy	Bert	Content	Average
Gossipcop	No Aug	90.48 ± 0.02	96.19 ± 0.07	95.73 ± 0.54	95.44 ± 0.72	94.46	63.19 ± 0.63	85.31 ± 1.73	89.01 ± 0.68	77.17 ± 0.76	78.67
	MAXPOOL	93.17 ± 0.87	96.52 ± 0.09	96.12 ± 0.13	95.79 ± 0.45	95.40	66.97 ± 1.06	86.28 ± 1.21	90.51 ± 0.35	78.98 ± 1.24	80.69
	GMT	93.58 ± 0.32	96.44 ± 0.07	96.25 ± 0.19	96.57 ± 0.02	95.71	75.37 ± 2.93	74.38 ± 0.02	72.39 ± 1.74	94.07 ± 1.11	79.05
	Type1	95.72 ± 0.27	96.84 ± 0.11	96.71 ± 0.26	96.96 ± 0.21	96.56	73.31 ± 1.22	87.58 ± 0.87	91.04 ± 1.25	93.31 ± 0.15	86.31
	Type2	94.11 ± 0.26	95.73 ± 0.24	96.24 ± 0.04	97.36 ± 0.14	95.86	71.77 ± 1.43	72.08 ± 1.22	76.78 ± 1.89	93.36 ± 1.88	78.50
	Type3	95.37 ± 0.33	96.64 ± 0.01	96.63 ± 0.09	96.79 ± 0.13	96.36	67.39 ± 1.65	87.72 ± 2.21	89.99 ± 1.88	83.86 ± 0.12	82.24
	Type4	95.14 ± 0.25	96.28 ± 0.21	94.43 ± 0.59	95.45 ± 0.11	95.33	90.24 ± 0.21	87.33 ± 0.88	77.64 ± 0.95	94.23 ± 1.13	87.36
	Type5	93.64 ± 0.35	96.06 ± 0.04	95.52 ± 0.02	97.04 ± 0.21	95.57	82.91 ± 0.66	86.07 ± 1.88	77.74 ± 0.87	94.17 ± 0.36	85.22
	Type6	93.83 ± 0.18	96.47 ± 0.06	96.58 ± 0.36	95.95 ± 0.25	95.71	80.88 ± 3.29	86.82 ± 1.31	72.99 ± 1.24	74.56 ± 2.75	78.81
	Type7	93.26 ± 0.06	96.25 ± 0.01	95.48 ± 0.04	95.34 ± 0.21	95.08	88.68 ± 3.23	80.66 ± 0.85	71.95 ± 1.18	95.02 ± 0.22	84.08
	Type8	92.12 ± 0.24	95.36 ± 0.13	92.45 ± 0.89	94.07 ± 0.88	93.50	83.61 ± 2.66	84.23 ± 0.72	75.09 ± 2.92	93.44 ± 1.73	84.10
	Type9	93.19 ± 0.05	96.47 ± 0.11	96.03 ± 0.03	95.94 ± 0.13	95.41	88.83 ± 0.16	82.88 ± 3.62	77.66 ± 1.01	94.76 ± 0.47	86.03
Politifact	No Aug	77.38 ± 1.81	79.55 ± 1.03	83.15 ± 0.23	84.89 ± 2.51	81.24	64.02 ± 1.59	73.45 ± 0.68	79.63 ± 1.91	81.02 ± 0.66	74.53
	MAXPOOL	77.82 ± 1.26	79.63 ± 0.79	83.49 ± 0.26	85.52 ± 1.22	81.61	66.74 ± 0.32	75.11 ± 0.72	80.09 ± 1.27	82.57 ± 0.95	76.13
	GMT	78.74 ± 0.98	81.46 ± 1.68	84.25 ± 0.81	81.54 ± 1.28	81.50	73.53 ± 4.79	60.27 ± 1.27	79.03 ± 4.61	81.23 ± 0.32	73.52
	Type1	78.96 ± 1.96	81.61 ± 2.49	83.94 ± 0.45	85.92 ± 1.35	82.61	69.01 ± 0.32	74.21 ± 1.12	78.36 ± 2.43	84.84 ± 0.31	76.60
	Type2	77.83 ± 1.64	77.41 ± 2.09	83.72 ± 0.74	85.07 ± 1.09	81.01	68.55 ± 0.96	64.93 ± 0.96	79.86 ± 0.32	82.81 ± 2.56	74.04
	Type3	78.73 ± 2.12	78.88 ± 3.41	83.03 ± 0.58	84.62 ± 1.52	81.31	72.85 ± 0.63	71.71 ± 0.95	80.54 ± 0.63	74.21 ± 1.92	74.83
	Type4	77.22 ± 0.94	81.22 ± 1.86	82.13 ± 3.72	83.37 ± 2.11	80.99	74.43 ± 1.61	75.56 ± 1.27	77.22 ± 3.79	81.67 ± 0.31	77.22
	Type5	77.38 ± 0.46	77.61 ± 1.83	82.24 ± 4.21	82.81 ± 1.27	80.01	71.71 ± 2.24	72.39 ± 0.64	76.41 ± 4.97	81.44 ± 1.27	75.49
	Type6	77.83 ± 1.19	79.31 ± 1.79	84.62 ± 0.98	82.35 ± 1.91	81.03	75.79 ± 0.96	73.75 ± 0.63	82.19 ± 1.45	75.56 ± 3.21	76.82
	Type7	77.08 ± 0.53	80.66 ± 2.85	84.16 ± 0.97	83.03 ± 1.59	81.23	71.49 ± 0.64	71.03 ± 1.28	79.05 ± 4.44	78.28 ± 1.29	74.96
	Type8	78.17 ± 1.49	77.95 ± 0.86	82.02 ± 1.97	83.57 ± 0.64	80.43	68.34 ± 0.87	66.96 ± 3.83	67.64 ± 1.59	74.88 ± 1.61	69.46
	Type9	78.28 ± 1.36	80.91 ± 1.97	84.39 ± 0.79	84.96 ± 1.29	82.14	76.91 ± 1.27	71.94 ± 1.26	77.37 ± 1.92	84.16 ± 0.63	77.60

Type 1 operator outperforms GMT by up to 2.14% on Gossipcop and 4.38% on Politifact, respectively. On average, for training with the GCN model, the Type 1 operator demonstrates the best performance under the NLP dataset. Moreover, maybe surprisingly, the gap between using the hybrid pooling operator and MAXPOOL can be up to 2.55% of clean accuracy (Gossipcop, Type1, Profile). Moving to the average improvement, we primarily focus on Type 1, Type 3, and Type 9, as they exhibit superior accuracy improvement compared to other hybrid pooling operators, as indicated in the Average column. In GCN model training on the NLP dataset, Type 1, Type 3, and Type 9 exhibit average accuracy improvements of up to 1.08%, 0.64%, and 0.35%, respectively, compared to MAXPOOL. On average, Type 1, Type 3, and Type 9 outperform GMT by up to 1.02%, 0.68%, and 0.45%, respectively. Overall, across both the maximum and average improvement values, Type 1 consistently outperforms other hybrid pooling operators.

The left part in Table 4 presents the results of the GAT model. Firstly, in both the Gossipcop-Profile and Politifact-Bert datasets, 4 out of the 9 hybrid pooling operators achieve higher test accuracy than MAXPOOL. Moreover, based on the results, the hybrid pooling operator (Type 3) brings the accuracy improvement by up to 2.31% than GMT on the Gossipcop-Spacy, and the operator Type 1 outperforms GMT by up to 2.16% on the Politifact-Content. On average, when training with the GAT model, the hybrid operator (Type 3) proves to be the optimal choice for the NLP dataset. It achieves the highest test accuracy on both the Gossipcop and Politifact datasets. We observe three different hybrid pooling operators, namely Type 1, Type 2, and Type 3, as they exhibit better accuracy improvement compared to others, as indicated in the Average column. Compared to MAXPOOL, Type 1, Type 2, and Type 3 exhibit average accuracy improvements of up to 0.59%, 0.30%, and 0.74%. Moving to GMT, Type 3 achieves the highest average accuracy improvement of 1.30%, followed by Type 1 at 1.26% and Type 2 at 0.91%. Overall, Type 3 shows the best performance in both maximum and average improvement analyses in GAT training.

The left part in Table 5 presents the results of the NLP dataset when training with the GraphSAGE model. Hybrid pooling operators can always help *Manifold-mixup* in improving test accuracy. For example, in evaluating the performance of hybrid pooling operators, we observe that 4 out of the 9 operators achieve better performance compared with MAXPOOL on both Gossipcop-Profile and Politifact-Bert. Interestingly, while operator Type 1 demonstrates the best performance in most cases, the results from the column Average indicate that operator Type 3 is the optimal choice for the Gossipcop dataset. Type 1 and Type 3 demonstrate the best performance, as indicated by the Average column. Compared to MAXPOOL, Type 1 performs an average accuracy improvement of 0.32%, which is slightly better than Type 3, at 0.30%. The same conclusion holds in the comparison with GMT, where Type 1 (1.50% on average) still outperforms Type 3 (1.32% on average).

The left part in Table 6 is the results of the GIN model. Surprisingly, in all cases examined, both operator Type 1 and operator Type 3 consistently outperform MAXPOOL on both the Gossipcop and Politifact datasets. Compared with GMT, the operator Type 1 improves the accuracy by up to 2.10% in Gossipcop-Profile and 1.58% on the Politifact-Bert, respectively. Also, results from the column Average show that the operator Type 1 is still the best choice for both the Gossipcop and Politifact datasets. We focus on Type 1, Type 2, and Type 3 for the average comparison, as the Average column indicates that these three types of hybrid pooling operators perform the best. Compared to MAXPOOL, Type 1 exhibits the highest average accuracy improvement, at up to 0.86%, followed by Type 2 at 0.39% and Type 3 at 0.32%. Moving to GMT, we reach the same conclusion as with MAXPOOL, where Type 1 achieves the highest average improvement, up to 0.97%, followed by Type 2 with 0.56% and Type 3 with 0.48%.

Table 4
Effectiveness of nine types of hybrid pooling w.r.t. test accuracy ↑ (average ± standard deviation, %) on original test data and robustness ↑ (average ± standard deviation, %) on robust test data of trained GAT models on NLP datasets. No Aug: without Manifold-Mixup data augmentation. A gray background highlights results that are better than MAXPOOL. The best average results are marked in red color.

		Test Accuracy					Robustness				
		Profile	Spacy	Bert	Content	Average	Profile	Spacy	Bert	Content	Average
Gossipcop	No Aug	91.19 ± 0.03	95.54 ± 0.02	96.49 ± 0.28	97.28 ± 0.43	95.13	90.69 ± 0.21	94.96 ± 0.06	94.49 ± 0.51	96.23 ± 0.63	94.09
	MAXPOOL	93.87 ± 0.47	96.00 ± 0.18	96.74 ± 0.41	97.39 ± 0.24	96.00	90.81 ± 0.52	95.22 ± 0.05	94.53 ± 1.77	96.31 ± 0.89	94.22
	GMT	92.38 ± 0.17	95.96 ± 0.01	95.93 ± 0.39	97.46 ± 0.08	95.43	86.33 ± 2.73	77.44 ± 2.01	79.04 ± 2.06	90.78 ± 3.81	83.40
	Type1	94.68 ± 0.01	96.56 ± 0.31	97.27 ± 0.21	97.54 ± 0.17	96.51	91.89 ± 0.79	95.71 ± 0.59	95.01 ± 2.35	96.82 ± 0.63	94.86
	Type2	94.76 ± 0.54	96.15 ± 0.43	96.99 ± 0.18	97.41 ± 0.09	96.33	91.02 ± 0.52	94.44 ± 1.21	94.95 ± 1.92	96.80 ± 0.49	94.30
	Type3	94.81 ± 0.13	97.08 ± 0.26	97.22 ± 0.16	97.53 ± 0.37	96.66	92.03 ± 0.76	96.46 ± 0.02	93.11 ± 0.91	96.22 ± 0.57	94.46
	Type4	92.57 ± 0.21	95.63 ± 0.02	94.96 ± 0.41	96.71 ± 0.08	94.97	90.68 ± 0.31	94.48 ± 1.03	93.87 ± 0.83	95.53 ± 0.07	93.64
	Type5	92.81 ± 0.59	95.98 ± 0.08	95.42 ± 0.32	97.05 ± 0.31	95.32	90.44 ± 0.46	94.13 ± 0.06	88.23 ± 1.36	95.82 ± 0.74	92.16
	Type6	95.18 ± 0.02	95.94 ± 0.69	96.63 ± 0.81	96.97 ± 0.07	96.18	91.85 ± 018	94.73 ± 0.26	92.75 ± 1.16	96.11 ± 1.19	93.86
	Type7	92.24 ± 0.07	95.23 ± 0.21	94.59 ± 0.04	96.66 ± 0.11	94.68	90.48 ± 0.26	94.07 ± 0.18	93.73 ± 0.95	94.83 ± 0.42	93.28
	Type8	91.92 ± 0.39	95.36 ± 0.47	95.46 ± 0.71	96.39 ± 0.48	94.78	90.24 ± 0.15	94.89 ± 0.41	94.41 ± 0.56	95.87 ± 0.52	93.85
	Type9	92.41 ± 0.09	95.78 ± 0.31	96.04 ± 0.31	97.15 ± 0.04	95.35	90.59 ± 0.11	95.25 ± 0.19	93.98 ± 2.14	96.54 ± 0.43	94.09
Politifact	No Aug	74.84 ± 1.38	79.19 ± 0.46	81.61 ± 2.88	85.29 ± 0.58	80.23	74.43 ± 0.32	77.01 ± 1.15	78.28 ± 1.92	84.16 ± 0.64	78.47
	MAXPOOL	75.57 ± 1.11	79.41 ± 0.96	82.58 ± 0.33	85.67 ± 2.14	80.81	75.26 ± 0.69	77.61 ± 0.77	79.54 ± 1.59	84.62 ± 012	79.26
	GMT	76.93 ± 0.46	80.32 ± 0.32	80.54 ± 1.22	83.71 ± 0.63	80.38	74.88 ± 2.87	71.72 ± 0.32	78.43 ± 1.82	81.14 ± 1.59	76.54
	Type1	75.11 ± 0.66	80.41 ± 1.05	83.04 ± 0.31	86.88 ± 0.64	81.36	74.45 ± 0.27	79.05 ± 1.21	81.89 ± 0.64	85.06 ± 1.28	80.11
	Type2	76.17 ± 0.69	79.86 ± 0.32	82.35 ± 0.95	85.52 ± 0.78	80.98	74.66 ± 0.63	78.73 ± 1.07	74.21 ± 0.61	82.57 ± 0.31	77.54
	Type3	78.73 ± 1.19	78.96 ± 0.33	82.21 ± 0.39	85.75 ± 0.98	81.41	76.02 ± 0.63	76.78 ± 0.19	78.27 ± 0.57	81.67 ± 0.32	78.19
	Type4	76.62 ± 2.03	79.04 ± 0.45	80.09 ± 0.22	83.26 ± 1.28	79.75	74.43 ± 1.24	77.59 ± 0.31	78.05 ± 0.96	75.34 ± 0.28	76.35
	Type5	76.61 ± 1.38	78.73 ± 0.53	79.76 ± 0.31	83.67 ± 1.45	79.69	75.46 ± 0.65	75.56 ± 1.92	75.29 ± 1.67	78.95 ± 2.15	76.32
	Type6	75.27 ± 0.94	78.96 ± 0.32	79.19 ± 0.29	83.81 ± 1.12	79.31	72.95 ± 1.26	76.02 ± 3.19	74.88 ± 2.87	81.21 ± 2.24	76.27
	Type7	76.69 ± 1.95	80.32 ± 0.96	81.99 ± 0.11	86.43 ± 0.77	81.36	75.33 ± 0.31	75.56 ± 3.83	79.64 ± 1.28	71.72 ± 0.33	75.56
	Type8	74.78 ± 2.77	79.19 ± 0.64	81.85 ± 0.21	85.35 ± 0.65	80.29	72.62 ± 1.23	75.06 ± 3.13	76.02 ± 2.55	78.95 ± 0.95	75.66
	Type9	75.08 ± 2.29	80.29 ± 1.08	81.02 ± 0.32	85.92 ± 0.81	80.58	74.45 ± 0.38	76.26 ± 3.75	79.63 ± 1.27	81.22 ± 2.87	77.89

Table 5
Effectiveness of nine types of hybrid pooling w.r.t. test accuracy ↑ (average ± standard deviation, %) on original test data and robustness ↑ (average ± standard deviation, %) on robust test data of trained GraphSAGE models on NLP datasets. No Aug: without Manifold-Mixup data augmentation. A gray background highlights results that are better than MAXPOOL. The best average results are marked in red color.

		Test Accuracy					Robustness				
		Profile	Spacy	Bert	Content	Average	Profile	Spacy	Bert	Content	Average
Gossipcop	No Aug	92.35 ± 0.02	96.53 ± 0.11	96.62 ± 0.02	97.58 ± 0.06	95.77	90.17 ± 0.37	94.68 ± 0.11	95.61 ± 0.32	96.34 ± 0.28	94.20
	MAXPOOL	93.65 ± 0.33	96.79 ± 0.09	96.81 ± 0.01	97.78 ± 0.02	96.26	90.72 ± 0.48	95.01 ± 0.02	95.84 ± 0.14	96.99 ± 0.03	94.64
	GMT	93.31 ± 0.02	94.61 ± 0.02	94.63 ± 0.09	96.71 ± 0.26	94.82	82.18 ± 2.74	84.72 ± 1.93	85.19 ± 2.24	93.31 ± 2.05	86.35
	Type1	94.41 ± 0.11	96.63 ± 0.15	96.88 ± 0.03	97.91 ± 0.03	96.46	94.05 ± 0.49	95.15 ± 0.05	88.89 ± 2.22	97.69 ± 0.09	93.95
	Type2	94.13 ± 0.12	96.48 ± 0.28	96.47 ± 0.18	97.81 ± 0.04	96.22	92.27 ± 0.83	90.08 ± 1.91	91.85 ± 3.39	97.26 ± 0.13	92.87
	Type3	94.81 ± 0.24	96.92 ± 0.07	96.98 ± 0.24	97.71 ± 0.09	96.61	94.24 ± 0.68	95.91 ± 0.09	95.94 ± 0.16	97.47 ± 0.05	95.89
	Type4	93.57 ± 0.55	94.69 ± 0.11	95.47 ± 0.68	96.66 ± 0.14	95.10	92.91 ± 0.51	94.23 ± 0.05	91.07 ± 1.08	92.57 ± 0.07	92.70
	Type5	92.88 ± 0.17	94.68 ± 0.49	95.36 ± 0.46	97.44 ± 0.22	95.09	92.18 ± 0.21	92.67 ± 2.19	93.78 ± 0.84	96.33 ± 0.92	93.74
	Type6	94.38 ± 0.23	95.32 ± 0.41	95.91 ± 0.06	97.73 ± 0.12	95.84	93.65 ± 0.33	94.17 ± 0.34	93.76 ± 2.16	96.64 ± 0.38	94.56
	Type7	92.61 ± 0.08	94.76 ± 0.35	94.05 ± 0.02	95.35 ± 0.08	94.19	91.82 ± 0.73	94.36 ± 0.02	90.99 ± 0.72	92.27 ± 0.04	92.36
	Type8	92.38 ± 0.42	95.71 ± 0.37	94.56 ± 0.56	96.13 ± 0.22	94.70	91.76 ± 0.62	94.43 ± 0.08	93.37 ± 0.06	94.77 ± 0.09	93.58
	Type9	93.12 ± 0.16	95.92 ± 0.18	94.90 ± 0.68	96.65 ± 0.32	95.15	92.45 ± 0.43	94.39 ± 0.06	92.35 ± 0.46	94.52 ± 0.19	93.43
Politifact	No Aug	77.65 ± 1.52	80.27 ± 0.99	82.53 ± 1.96	83.84 ± 1.32	81.07	76.47 ± 0.45	78.05 ± 0.95	80.31 ± 1.59	81.91 ± 1.28	79.19
	MAXPOOL	77.83 ± 0.64	80.54 ± 0.72	83.26 ± 0.96	85.37 ± 0.88	81.75	76.82 ± 0.42	78.51 ± 0.96	82.12 ± 0.31	82.12 ± 0.31	79.89
	GMT	77.15 ± 0.33	79.19 ± 1.25	84.01 ± 0.94	84.11 ± 1.05	81.12	71.94 ± 0.57	65.61 ± 3.21	81.44 ± 0.79	78.06 ± 0.95	74.26
	Type1	78.89 ± 1.88	80.69 ± 0.95	83.71 ± 0.45	86.27 ± 0.94	82.39	77.37 ± 0.63	79.86 ± 0.33	81.67 ± 2.23	85.29 ± 0.91	81.05
	Type2	77.61 ± 0.32	80.59 ± 1.19	82.13 ± 1.59	84.73 ± 2.23	81.27	76.94 ± 1.19	76.46 ± 0.64	81.21 ± 0.32	82.35 ± 2.07	79.24
	Type3	78.73 ± 0.67	80.43 ± 1.17	82.96 ± 0.53	84.71 ± 3.31	81.71	75.33 ± 2.24	79.18 ± 0.63	81.89 ± 0.65	82.34 ± 1.26	79.69
	Type4	77.23 ± 0.55	80.66 ± 0.67	84.01 ± 0.94	83.76 ± 0.02	81.42	75.79 ± 0.33	79.41 ± 0.96	80.99 ± 2.56	74.88 ± 0.33	77.77
	Type5	76.89 ± 0.99	80.32 ± 0.89	82.58 ± 1.23	81.86 ± 0.12	80.41	75.56 ± 1.27	74.66 ± 1.29	80.84 ± 1.46	76.71 ± 3.52	76.94
	Type6	76.57 ± 0.23	80.09 ± 0.63	83.27 ± 1.21	83.26 ± 0.22	80.80	73.98 ± 1.59	76.69 ± 2.24	79.18 ± 1.92	78.28 ± 2.56	77.03
	Type7	76.93 ± 1.28	79.37 ± 1.27	82.21 ± 1.14	82.81 ± 0.11	80.33	71.72 ± 0.32	78.28 ± 0.61	78.28 ± 0.61	73.53 ± 0.85	75.45
	Type8	76.71 ± 0.97	80.55 ± 1.46	81.91 ± 1.28	84.39 ± 0.33	80.89	74.21 ± 1.17	78.96 ± 0.31	78.96 ± 0.31	75.86 ± 2.14	77.00
	Type9	76.69 ± 0.32	80.47 ± 1.02	84.16 ± 0.78	85.31 ± 0.32	81.66	75.56 ± 0.66	79.63 ± 0.67	83.48 ± 0.28	76.92 ± 3.19	78.90

Table 6
Effectiveness of nine types of hybrid pooling w.r.t. test accuracy ↑ (average ± standard deviation, %) on original test data and robustness ↑ (average ± standard deviation, %) on robust test data of trained GIN models on NLP datasets. No Aug: without Manifold-Mixup data augmentation. A gray background highlights results that are better than MAXPOOL. The best average results are marked in red color.

		Test Accuracy					Robustness				
		Profile	Spacy	Bert	Content	Average	Profile	Spacy	Bert	Content	Average
Gossipcop	No Aug	93.39 ± 0.29	95.86 ± 0.34	94.32 ± 0.11	95.75 ± 0.45	94.83	82.04 ± 1.99	86.81 ± 3.04	72.92 ± 1.04	74.18 ± 0.16	78.99
	MAXPOOL	93.51 ± 0.27	96.01 ± 0.24	94.91 ± 0.56	96.42 ± 0.46	95.21	85.46 ± 0.83	87.01 ± 3.16	74.27 ± 2.87	76.45 ± 0.22	80.80
	GMT	92.44 ± 0.25	95.53 ± 0.28	94.73 ± 0.13	96.29 ± 0.03	94.75	70.63 ± 1.22	75.65 ± 3.64	73.23 ± 0.74	84.51 ± 0.91	76.01
	Type1	94.54 ± 0.21	96.51 ± 0.22	95.41 ± 0.07	97.19 ± 0.19	95.91	87.98 ± 0.48	88.36 ± 2.35	92.04 ± 0.15	92.65 ± 0.27	90.26
	Type2	94.06 ± 0.32	95.83 ± 0.18	95.11 ± 0.38	96.89 ± 0.11	95.47	72.01 ± 0.18	72.79 ± 0.35	71.02 ± 0.22	72.06 ± 2.27	71.97
	Type3	93.83 ± 0.34	96.17 ± 0.31	94.91 ± 0.71	96.62 ± 0.25	95.38	81.04 ± 0.25	76.13 ± 5.34	71.11 ± 0.98	85.57 ± 2.41	78.46
	Type4	93.42 ± 0.22	95.92 ± 0.29	93.12 ± 0.69	96.75 ± 0.31	94.80	84.47 ± 0.74	73.31 ± 1.97	83.46 ± 2.16	86.18 ± 2.31	81.86
	Type5	93.22 ± 0.06	95.05 ± 0.37	91.98 ± 1.22	96.94 ± 0.41	94.30	91.68 ± 0.76	74.59 ± 1.47	84.05 ± 4.13	88.59 ± 1.97	84.73
	Type6	94.66 ± 0.52	96.11 ± 0.33	94.99 ± 0.09	96.61 ± 0.76	95.59	93.86 ± 0.22	75.95 ± 0.66	78.17 ± 3.36	89.11 ± 2.08	84.27
	Type7	93.17 ± 0.35	95.73 ± 0.23	94.13 ± 0.79	96.96 ± 0.15	95.00	73.38 ± 2.57	85.99 ± 2.58	81.51 ± 2.49	89.68 ± 0.75	82.64
	Type8	92.02 ± 0.86	95.43 ± 0.45	92.05 ± 0.73	95.44 ± 0.26	93.74	62.14 ± 1.22	86.01 ± 0.44	71.77 ± 0.11	65.76 ± 4.51	71.42
	Type9	93.08 ± 0.34	95.93 ± 0.21	93.81 ± 0.41	97.01 ± 0.51	94.96	76.69 ± 1.41	83.78 ± 0.72	74.36 ± 2.96	90.95 ± 1.47	81.45
Politifact	No Aug	77.01 ± 1.91	80.43 ± 0.68	84.16 ± 0.32	84.73 ± 1.35	81.58	61.99 ± 1.92	76.24 ± 0.32	81.89 ± 0.45	50.75 ± 0.17	67.72
	MAXPOOL	77.33 ± 0.89	81.68 ± 1.14	84.35 ± 0.55	84.88 ± 1.59	82.06	68.78 ± 0.63	77.17 ± 0.98	82.12 ± 0.31	51.12 ± 0.71	69.80
	GMT	77.74 ± 1.26	81.75 ± 1.01	84.39 ± 1.69	84.95 ± 1.35	82.21	61.68 ± 3.21	61.43 ± 0.79	83.48 ± 0.32	82.81 ± 1.93	72.35
	Type1	78.28 ± 0.93	82.24 ± 0.57	85.97 ± 0.27	85.41 ± 1.19	82.98	73.31 ± 2.56	76.91 ± 1.27	83.03 ± 0.28	50.93 ± 0.08	71.05
	Type2	78.06 ± 1.61	81.46 ± 0.28	85.52 ± 0.35	85.04 ± 1.77	82.52	74.21 ± 0.64	74.36 ± 1.59	81.67 ± 0.35	50.61 ± 0.15	70.21
	Type3	77.92 ± 2.85	81.87 ± 0.94	85.07 ± 0.31	85.24 ± 1.36	82.53	75.06 ± 3.78	76.46 ± 0.64	80.99 ± 1.17	50.82 ± 0.43	70.83
	Type4	77.74 ± 0.74	80.09 ± 0.55	84.39 ± 0.96	83.86 ± 1.88	81.52	75.56 ± 1.92	76.69 ± 2.25	82.65 ± 1.39	79.86 ± 2.88	78.69
	Type5	77.64 ± 1.72	80.77 ± 0.98	84.46 ± 0.53	82.81 ± 0.64	81.42	76.41 ± 0.36	78.28 ± 0.63	83.71 ± 0.65	78.05 ± 0.96	79.11
	Type6	78.05 ± 1.08	81.45 ± 0.64	84.16 ± 0.22	84.05 ± 1.86	81.93	74.66 ± 0.79	77.92 ± 3.48	82.36 ± 0.67	77.31 ± 4.35	78.06
	Type7	77.83 ± 1.61	80.03 ± 0.49	84.44 ± 0.68	84.39 ± 0.94	81.67	75.11 ± 0.66	76.92 ± 1.92	81.89 ± 1.28	76.79 ± 4.29	77.68
	Type8	78.39 ± 0.78	80.01 ± 0.51	83.81 ± 1.32	81.61 ± 0.52	80.96	57.19 ± 2.36	67.64 ± 2.87	77.14 ± 1.61	61.24 ± 3.87	65.80
	Type9	77.74 ± 1.71	80.99 ± 0.68	84.77 ± 1.03	85.19 ± 1.93	82.17	74.36 ± 2.28	77.61 ± 0.92	83.48 ± 0.37	83.03 ± 2.23	79.62

Table 7 presents the results of PL datasets. We can see that the advanced graph pooling operators, including GMT and hybrid pooling operators, are more helpful than the MAXPOOL for *Manifold-Mixup* when dealing with PL data, where the best results are always from the hybrid pooling operators. Similar to the results of traditional NLP datasets, the Type 1 hybrid pooling operator outperforms MAXPOOL in most situations (6 out of 8 cases). Unlike the NLP datasets, the operator Type 3 consistently achieves better results than MAXPOOL. In PL datasets, the gap between using GMT and hybrid pooling operators can be up to 2.36% of clean accuracy (GIN-Virtual, Python800, Type 1). When considering average improvement, we focus on Type 1 and Type 3 across all GNNs, as they demonstrate better performance compared to others in the results. Compared to MAXPOOL, Type 1 exhibits a better average accuracy improvement of 0.59% compared to Type 3, which is 0.44%. In the case of GMT, Type 1 also outperforms Type 3 in average accuracy improvement, with 0.92% compared to 0.90%.

Statistical analysis. To check the significance of the impact of hybrid pooling on the model accuracy, we conduct a statistical analysis. Specifically, we use *Wilcoxon signed-rank test* (Woolson, 2007) to analyze the significance. Since there are many comparisons, we adapt *Bonferroni correction* (Armstrong, 2014) to adjust the *P-values* before concluding. The upper component in Table 8 presents the comparison between pooling operators and No Aug, MAXPOOL, and GMT. From the results, we can see that Type1 significantly outperforms the three baselines both on PL and NLP. In most cases, hybrid pooling operators such as Type 4, Type 5, Type 6, and Type 9 are significantly worse than the baselines.

Overall, based on the analysis including maximum improvement, average improvement, and statistical significance, we can conclude that hybrid pooling operators have a significant impact on the effectiveness of the Mixup data augmentation technique when dealing with graph-structured datasets. More specifically, for accuracy improvement, Type 1 ($\mathcal{M}_{sum}(\mathcal{P}_{att}, \mathcal{P}_{max})$) is the best choice among our considered candidates, while Type 3 ($\mathcal{M}_{concat}(\mathcal{P}_{att}, \mathcal{P}_{max})$) also demonstrates relatively strong performance. Both Type 1 and Type 3 are recommended for use in graph classification tasks, given their superior performance across all evaluated metrics, datasets (PL and NLP), and GNN architectures.

Answer to RQ1: Hybrid pooling operators have a significant impact in enhancing the effectiveness of *Manifold-Mixup*-based graph learning in terms of clean accuracy. Considering the analysis including maximum improvement, average improvement, and statistical significance, we recommend Type 1 as the optimal choice. Specifically, compared to GMT, the Type 1 operator ($\mathcal{M}_{sum}(\mathcal{P}_{att}, \mathcal{P}_{max})$) demonstrates up to 4.38% accuracy improvements on the NLP dataset and 2.36% on the PL dataset, respectively.

5.2. RQ2: How effective are hybrid pooling operators for enhancing the robustness of mixup-based graph learning?

Robustness analysis. The right part in Table 3 presents the results of the robustness of the GCN model on the NLP dataset. Firstly, it is observed that *Manifold-Mixup* significantly enhances the generalization ability of the GNN model. In addition, although GMT is an advanced graph pooling operator, it is unable to help *Manifold-Mixup* effectively enhance the robustness of the GCN model. It proves to be effective in only 3 out of 8 cases. Then, concerning hybrid pooling operators, surprisingly, all operators obtain better performance in improving the robustness than MAXPOOL on the Gossipcop-Profile and Politifact-Profile datasets. Furthermore, in terms of robustness, the operator Type 9 outperforms GMT by a significant margin of up to 13.46% (Gossipcop-Profile), and the operator Type 4 obtains an even higher improvement in robustness by up to 15.29% (Politifact-Spacy) compared to GMT. When examining the average robustness improvement, we focus on three hybrid pooling operators including Type 1, Type 4, and Type 9, as they demonstrate better performance compared to others, as shown in the Average column. Compared to MAXPOOL, Type 9 exhibits the highest average robustness improvement, at 6.18%, followed by Type 4 at 5.96% and Type 1 at 3.38%. In the case of GMT, different from MAXPOOL, Type 4 achieves a better average robustness improvement of 6.23% compared to Type 1 at 6.18% and Type 9 at 5.74%.

The right part in Table 4 shows the result of the GAT model. Surprisingly, all of the cases show that GMT can be unable to help *Manifold-Mixup* improve the robustness compared to MAXPOOL. In contrast, 4 out of the 9 hybrid operators on the Gossipcop-Profile and 3 out of the 9 hybrid operators on the Politifact-Bert are able to achieve better performance than MAXPOOL. In particular, the operator Type 3 improves the performance of robustness by up to 19.02% on the Gossipcop-Spacy compared to GMT. The operator Type 1 outperforms GMT by up to 7.33% on the Politifact-Spacy. However, the results from the column Average show that the operator Type 1 is still the optimal choice for GAT in robustness improvement. We focus on Type 1 and Type 3 for the analysis of average robustness improvement, as these two hybrid pooling operators outperform others, as indicated by the Average column. Type 1 exhibits a higher average robustness improvement of 0.86% compared to Type 3 with 0.41% when compared to MAXPOOL. Compared to GMT, we can reach a similar conclusion that Type 1 exhibits the best improvement in GAT robustness, with an average improvement of 7.57%, whereas Type 3 only shows an average improvement of 5.67%.

The right part in Table 5 presents the robustness results of the GraphSAGE model. Similar to the finding from Table 4, GMT is unable to help *Manifold-Mixup* in robustness improvement. However, all hybrid operators are effective in improving the robustness compared to MAXPOOL on the Gossipcop-Profile. Compared to GMT, the operator Type 1 improves the robustness by up to 11.87% on the Gossipcop-Profile and 14.25% on the Politifact-Spacy. Similar to GAT, we continue to focus on Type 1 and Type 3 as these two hybrid pooling operators demonstrate the best performance compared to others, as indicated by the Average column. When compared to MAXPOOL, Type 1 demonstrates a superior average robustness improvement of 1.16%, while Type 3 exhibits an average improvement of 0.73%. Different from the comparison in MAXPOOL, Type 3 exhibits a superior average robustness improvement of 8.71% compared to Type 1, which achieves 7.19%, when considering GMT.

The right part in Table 6 presents the results of the GIN model. Maybe surprisingly, the gap between MAXPOOL and GMT can reach 31.69% (Politifact-Content) in terms of robustness. Moving to robustness improvement for GIN training, the operator Type 6 outperforms GMT by up to 23.23% on the Gossipcop-Profile. Similarly, the Type 5 operator achieves an improvement of up to 16.49% compared to GMT on the Politifact-Spacy. We analyze the average robustness improvement of Type 1 and Type 9 as they demonstrate superior performance compared to others, based on the Average column. Type 9 exhibits a superior average robustness improvement of 6.74% compared to MAXPOOL, while Type 1 achieves a lower average improvement of 5.41%. However, moving to GMT, Type 1 demonstrates a better average robustness improvement of 10.52% compared to Type 9, which only achieves 6.36%.

Table 7 presents the results of JAVA250 and Python800. Hybrid pooling operators are more effective in improving the robustness of GNN models than GMT and MAXPOOL. Notably, the operator Type 1 achieves the highest performance in 4 out of 8 cases. Moving to the gap between hybrid operators and GMT, the operator Type 1 can outperform GMT by up to 10.32% in the robustness improvement (GCN, Python800, Type 1). We analyze the average robustness improvement in the PL dataset using Type 1, Type 3, and Type 9. These three hybrid pooling operators are selected based on their superior performance compared to others, as indicated by the results from the Average column. When compared to MAXPOOL, Type 1 exhibits the highest average robustness improvement at 5.55%, followed by Type 3

Table 7
Effectiveness of nine types of hybrid pooling w.r.t. test accuracy ↑ (average ± standard deviation, %) on original test data and robustness ↑ (average ± standard deviation, %) on robust test data on PL datasets. No Aug: without Manifold-Mixup data augmentation. A gray background highlights results that are better than MAXPOOL. The best average results are marked in red color.

		Test Accuracy		Robustness			Test Accuracy		Robustness	
		JAVA250	Python800	JAVA250	Python800		JAVA250	Python800	JAVA250	Python800
GCN	No Aug	92.29 ± 0.17	93.73 ± 0.03	82.05 ± 2.03	82.37 ± 1.65	GCN-Virtual	93.46 ± 0.07	93.89 ± 0.18	85.01 ± 1.34	85.48 ± 1.26
	MAXPOOL	92.39 ± 0.08	93.94 ± 0.07	82.23 ± 2.14	82.63 ± 1.32		93.84 ± 0.31	94.27 ± 0.21	85.66 ± 2.01	86.13 ± 1.34
	GMT	92.66 ± 0.32	94.36 ± 0.21	82.69 ± 1.78	82.04 ± 2.34		92.49 ± 0.26	93.88 ± 0.33	84.17 ± 1.87	82.77 ± 1.87
	Type1	92.89 ± 0.12	94.29 ± 0.06	86.85 ± 1.67	**92.36 ± 1.02**		**94.56 ± 0.27**	**95.02 ± 0.32**	**91.19 ± 1.53**	**92.42 ± 1.75**
	Type2	92.58 ± 0.25	93.86 ± 0.02	86.43 ± 2.07	86.81 ± 2.46		93.32 ± 0.34	94.18 ± 0.39	88.31 ± 2.01	89.14 ± 2.01
	Type3	**93.72 ± 0.15**	**94.99 ± 0.12**	**91.27 ± 1.43**	90.94 ± 1.89		94.06 ± 0.28	94.43 ± 0.08	86.54 ± 1.26	90.56 ± 1.09
	Type4	90.68 ± 0.21	94.02 ± 0.09	82.49 ± 2.11	84.38 ± 2.11		91.21 ± 0.23	94.39 ± 0.28	85.11 ± 1.34	88.15 ± 1.54
	Type5	91.19 ± 0.13	93.98 ± 0.08	83.62 ± 2.02	83.15 ± 1.67		91.04 ± 0.31	93.78 ± 0.11	84.38 ± 1.76	85.31 ± 1.76
	Type6	91.69 ± 0.16	93.57 ± 0.04	83.68 ± 1.25	83.77 ± 2.01		92.17 ± 0.31	93.54 ± 0.13	84.16 ± 2.11	83.59 ± 2.11
	Type7	92.78 ± 0.08	93.09 ± 0.21	82.01 ± 1.67	82.17 ± 1.47		92.35 ± 0.29	92.89 ± 0.25	83.04 ± 1.98	81.37 ± 1.55
	Type8	91.96 ± 0.19	92.45 ± 0.19	81.98 ± 1.87	81.89 ± 1.88		92.69 ± 0.45	92.43 ± 0.19	82.71 ± 2.31	81.04 ± 1.65
	Type9	92.81 ± 0.05	93.66 ± 0.12	84.74 ± 2.56	84.87 ± 2.01		93.31 ± 0.33	93.29 ± 0.21	84.35 ± 1.78	84.19 ± 1.34
GIN	No Aug	92.57 ± 0.04	93.94 ± 0.26	84.07 ± 1.43	82.12 ± 1.86	GIN -Virtual	91.46 ± 0.66	94.06 ± 0.08	83.12 ± 1.46	84.37 ± 1.28
	MAXPOOL	93.35 ± 0.05	94.15 ± 0.03	85.11 ± 2.02	82.26 ± 1.65		92.57 ± 0.36	94.46 ± 0.16	84.17 ± 2.11	85.69 ± 2.13
	GMT	93.41 ± 0.13	94.27 ± 0.18	83.14 ± 1.35	83.81 ± 2.09		92.36 ± 0.49	93.01 ± 0.22	83.92 ± 1.86	85.37 ± 1.24
	Type1	93.13 ± 0.11	**95.11 ± 0.08**	85.65 ± 2.07	87.92 ± 1.83		**93.08 ± 0.45**	**95.37 ± 0.12**	**90.89 ± 1.22**	91.03 ± 2.19
	Type2	93.11 ± 0.17	94.31 ± 0.13	85.01 ± 1.65	88.38 ± 1.76		92.47 ± 0.23	94.89 ± 0.14	87.13 ± 2.01	89.87 ± 1.22
	Type3	93.67 ± 0.08	94.73 ± 0.11	85.72 ± 2.11	**92.47 ± 1.38**		92.98 ± 0.28	95.02 ± 0.09	85.92 ± 2.12	90.11 ± 1.76
	Type4	93.21 ± 0.19	93.97 ± 0.09	84.83 ± 1.27	86.71 ± 2.06		92.03 ± 0.14	93.88 ± 0.17	84.32 ± 1.23	86.37 ± 2.18
	Type5	93.42 ± 0.04	93.12 ± 0.15	84.66 ± 2.01	84.32 ± 2.11		91.78 ± 0.65	93.43 ± 0.21	83.27 ± 1.39	85.43 ± 1.23
	Type6	**94.81 ± 0.12**	94.22 ± 0.12	**91.43 ± 1.33**	85.49 ± 1.68		92.47 ± 0.28	94.35 ± 0.11	83.09 ± 2.87	84.67 ± 2.17
	Type7	93.14 ± 0.15	93.73 ± 0.05	82.89 ± 1.32	82.39 ± 1.88		92.51 ± 0.12	92.79 ± 0.19	82.37 ± 2.18	83.03 ± 1.75
	Type8	92.08 ± 0.22	93.42 ± 0.14	81.31 ± 1.43	81.57 ± 2.02		91.98 ± 0.29	92.13 ± 0.21	82.11 ± 1.77	82.34 ± 1.58
	Type9	93.63 ± 0.07	93.89 ± 0.06	83.47 ± 2.13	83.59 ± 1.22		92.39 ± 0.39	93.43 ± 0.23	83.47 ± 1.08	**92.58 ± 1.66**

Table 8
Results of statistical testing on Accuracy and Robustness. True: indicates that the comparison is statistically significant after the *Bonferroni correction* adjustment. Statistical testing methods: *Wilcoxon signed-rank test*. GNNs(PL): represents the statistical testing conducted on all GNN models including GCN, GIN, GCN-Virtual, and GIN-Virtual. A gray background highlights the result marked as True.

Test Accuracy

	GCN (NLP)			GAT (NLP)			GraphSAGE (NLP)			GIN (NLP)			GNNs (PL)		
	No Aug	MAXPOOL	GMT	No Aug	MAXPOOL	GMT	No Aug	MAXPOOL	GMT	No Aug	MAXPOOL	GMT	No Aug	MAXPOOL	GMT
Type1	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
Type2	True	–	–	True	–	–	–	–	–	True	–	True	True	–	–
Type3	–	–	–	True	–	True	True	–	True	True	True	True	True	True	
Type4	–	–	–	–	True	True	–	–	–	–	–	–	–	True	–
Type5	–	–	–	–	True	–	–	True	–	–	–	–	–	True	True
Type6	–	–	–	–	–	–	–	True	–	True	–	–	–	–	–
Type7	–	–	–	–	–	–	True	True	True	–	–	–	–	True	True
Type8	–	True	True	–	True	–	–	True	–	–	True	True	True	True	True
Type9	True	–	–	–	–	–	–	True	–	–	–	–	–	–	–

Robustness

Type1	True	–	–	True	True	True	–	–	True	True	True	–	True	True	True
Type2	–	–	–	–	–	True	–	–	True	–	–	–	True	True	True
Type3	–	–	–	–	–	True	True	–	True	–	–	–	True	True	True
Type4	–	–	True	True	True	–	–	–	–	–	–	–	True	–	True
Type5	–	–	–	True	True	True	–	True	True	True	–	–	–	–	True
Type6	–	–	–	True	True	True	–	–	True	True	–	–	–	–	-
Type7	–	–	–	–	True	–	True	True	–	–	–	True	True	True	True
Type8	–	–	–	True	True	–	–	True	–	–	True	–	True	True	True
Type9	–	–	True	–	–	True	–	–	True	–	–	True	–	–	–

at 4.96% and Type 9 at 1.62%. Moving to GMT, Type 1 still shows the highest improvement at 6.30%, followed by Type 3 at 5.70% and Type 9 at 1.75%.

Statistical analysis. We also conduct statistical analysis to check the significance of the robustness improvement by using hybrid pooling. The lower part in Table 8 presents the results. We can see that similar to the results of test accuracy, Type 1 significantly outperforms No Aug, MAXPOOL, and GMT in 10 out of 15 cases. This indicates that Type 1 is a relatively stable way to enhance the robustness of GNN models. Besides, we find that most hybrid pooling operators show better robustness improvement on GAT in NLP and all GNNs in PL, compared to No Aug, MAXPOOL, and GMT.

Overall, based on the analysis of NLP and PL datasets, we find that hybrid pooling operators can indeed improve the robustness of GNNs, whether compared to MAXPOOL or GMT, across all GNNs. We comprehensively assess the performance of all hybrid operators across these three evaluation metrics based on maximum improvement, average improvement, and statistical significance. Our analysis concludes that Type 1 exhibits the highest effectiveness, closely followed by Type 3.

Answer to RQ2: Hybrid pooling operators can enhance the robustness of GNN models. Among our studied operators, based on the analysis of improvement, average enhancement, and statistical significance, we recommend Type 1 and Type 3 for improving the robustness of GNNs. Concretely, the operator Type 1 ($\mathcal{M}_{sum}(\mathcal{P}_{att}, \mathcal{P}_{max})$) improves the robustness by 10.23% compared to GMT on the PL dataset.

5.3. RQ3: How does the hyperparameter setting affect the effectiveness of manifold-mixup when hybrid pooling operators are applied?

From the last part, we know that hybrid pooling operators are more helpful for *Manifold-Mixup* when dealing with graph-structured data. However, the potential influence of α that is introduced in Section 3.1, which is the key hyperparameter in the Mixup technique (Zhang et al., 2018a; Dong et al., 2023a; Verma et al., 2019; Yun et al., 2019), on the effectiveness of the Mixup training is still unclear. In this part, we tend to explore such influence using the dataset Politifact with Profile node embedding and set the α from 0.05 to 0.5 in 0.05 intervals.

Table 9
Results (average ± standard deviation, %) of trained GCN models with *Manifold-Mixup* (dataset: Politifact) using hybrid pooling operators with different α settings. The best results are marked in red color.

	Test Accuracy											
Mixup Ratio Alpha	**0.05**	**0.10**	**0.15**	**0.20**	**0.25**	**0.30**	**0.35**	**0.40**	**0.45**	**0.50**	Average	Std
Type1	79.07 ± 0.57	78.96 ± 1.96	77.01 ± 1.21	78.73 ± 0.46	78.28 ± 0.45	79.49 ± 0.94	77.83 ± 0.91	78.28 ± 1.57	77.68 ± 0.26	75.87 ± 1.45	78.12	1.08
Type2	75.79 ± 0.32	77.83 ± 1.64	74.36 ± 1.14	75.57 ± 2.39	76.32 ± 1.14	75.42 ± 2.09	76.17 ± 0.52	77.06 ± 1.31	78.13 ± 0.69	75.11 ± 0.91	76.18	1.20
Type3	76.02 ± 1.28	78.73 ± 2.12	77.38 ± 2.39	76.77 ± 1.83	77.22 ± 1.88	73.75 ± 3.59	78.58 ± 1.38	73.15 ± 2.57	76.47 ± 2.75	77.38 ± 0.92	76.55	1.84
Type4	77.15 ± 0.31	77.22 ± 0.94	76.92 ± 0.46	76.32 ± 0.94	77.52 ± 1.04	77.53 ± 0.26	77.98 ± 0.69	77.38 ± 0.91	76.92 ± 0.91	76.17 ± 1.31	77.11	**0.55**
Type5	77.22 ± 1.38	77.38 ± 0.46	77.53 ± 1.13	77.32 ± 1.73	76.17 ± 3.77	73.61 ± 1.14	76.02 ± 2.75	76.47 ± 0.45	75.26 ± 2.65	77.38 ± 2.75	76.44	1.25
Type6	78.06 ± 0.95	77.83 ± 1.19	75.72 ± 0.26	76.17 ± 0.26	76.47 ± 0.91	75.87 ± 1.16	75.56 ± 1.19	75.56 ± 0.78	77.22 ± 0.94	76.92 ± 0.89	76.54	0.93
Type7	77.83 ± 1.28	77.08 ± 0.53	77.07 ± 1.14	77.98 ± 0.94	77.61 ± 1.41	76.47 ± 1.21	77.08 ± 0.53	77.98 ± 1.38	77.68 ± 0.26	75.72 ± 2.49	77.25	0.73
Type8	78.84 ± 0.77	78.17 ± 1.49	79.64 ± 0.64	77.37 ± 1.99	79.34 ± 1.13	79.78 ± 0.94	79.04 ± 1.45	79.34 ± 0.26	78.58 ± 0.92	79.79 ± 1.31	**78.99**	0.78
Type9	77.38 ± 0.64	78.28 ± 1.36	77.08 ± 0.14	78.28 ± 0.45	77.98 ± 0.94	77.83 ± 1.21	78.58 ± 0.69	77.98 ± 0.94	77.53 ± 1.13	78.43 ± 0.69	77.94	0.49
Average	77.48	**77.94**	76.97	77.17	77.43	76.64	77.43	77.02	77.27	76.97	77.23	0.98
	Robustness											
Type1	70.78 ± 0.45	69.01 ± 0.32	68.33 ± 0.21	68.94 ± 0.69	68.63 ± 0.93	68.74 ± 0.45	67.82 ± 0.54	67.91 ± 0.34	67.58 ± 0.54	67.23 ± 1.55	68.50	1.00
Type2	68.97 ± 0.65	68.55 ± 0.96	68.21 ± 1.34	68.44 ± 1.29	69.01 ± 0.74	68.96 ± 1.99	68.97 ± 0.16	68.96 ± 0.91	68.98 ± 0.86	68.65 ± 0.86	68.77	0.29
Type3	73.04 ± 0.55	72.85 ± 0.63	71.34 ± 1.23	71.03 ± 0.63	71.28 ± 0.58	70.35 ± 2.39	71.98 ± 2.58	70.25 ± 1.34	70.64 ± 1.45	70.32 ± 1.23	71.31	1.02
Type4	74.54 ± 1.32	74.43 ± 1.61	74.12 ± 1.56	74.01 ± 1.45	74.86 ± 0.94	74.59 ± 1.16	74.62 ± 0.94	74.28 ± 0.51	74.03 ± 3.21	73.99 ± 2.11	74.35	**0.31**
Type5	72.57 ± 0.87	71.71 ± 2.24	71.75 ± 0.98	71.61 ± 0.76	70.97 ± 2.43	70.01 ± 0.54	70.99 ± 1.45	70.78 ± 0.94	70.16 ± 1.05	70.19 ± 1.87	71.07	0.83
Type6	77.13 ± 1.03	75.79 ± 0.96	75.01 ± 0.87	75.33 ± 1.44	75.41 ± 1.92	75.02 ± 0.67	75.01 ± 0.55	74.98 ± 1.38	75.22 ± 1.24	75.11 ± 0.65	75.40	0.66
Type7	71.87 ± 0.83	71.49 ± 0.64	71.37 ± 0.56	71.89 ± 0.86	71.81 ± 1.46	71.45 ± 1.96	71.01 ± 0.69	71.01 ± 2.31	70.78 ± 0.06	71.56 ± 1.66	71.42	0.39
Type8	68.55 ± 0.87	68.34 ± 0.87	70.23 ± 1.24	69.87 ± 1.79	70.12 ± 2.08	70.45 ± 0.45	70.08 ± 3.02	70.11 ± 1.45	70.01 ± 0.91	70.12 ± 0.86	69.79	0.72
Type9	76.17 ± 0.32	76.91 ± 1.27	76.08 ± 1.44	76.45 ± 0.95	76.29 ± 1.49	76.13 ± 1.81	76.58 ± 2.09	76.21 ± 1.45	76.03 ± 1.23	75.31 ± 2.01	**76.22**	0.42
Average	**72.62**	72.12	71.83	71.95	72.04	71.74	71.90	71.61	71.49	71.39	71.87	0.63

Table 10
Results (average ± standard deviation, %) of trained GAT models with *Manifold-Mixup* (dataset: Politifact) using hybrid pooling operators with different α settings. The best results are marked in red color.

	Test Accuracy											
Mixup Ratio Alpha	**0.05**	**0.10**	**0.15**	**0.20**	**0.25**	**0.30**	**0.35**	**0.40**	**0.45**	**0.50**	Average	Std
Type1	76.32 ± 1.72	75.11 ± 0.66	75.27 ± 0.94	76.47 ± 1.36	76.02 ± 0.45	75.11 ± 0.79	75.26 ± 0.69	76.17 ± 1.05	74.96 ± 1.38	77.23 ± 1.14	75.79	0.76
Type2	76.77 ± 0.94	76.17 ± 0.69	76.17 ± 1.83	76.47 ± 1.21	74.96 ± 2.23	74.51 ± 0.94	76.17 ± 2.27	77.22 ± 1.31	76.47 ± 1.57	75.56 ± 0.79	76.05	0.82
Type3	77.68 ± 1.14	78.73 ± 1.19	77.68 ± 2.04	77.83 ± 1.19	77.53 ± 1.46	77.83 ± 0.45	78.73 ± 0.91	77.23 ± 0.69	77.98 ± 2.57	77.83 ± 0.45	77.91	0.48
Type4	76.77 ± 1.15	76.62 ± 2.03	75.57 ± 2.39	76.32 ± 2.23	77.37 ± 1.19	76.47 ± 3.14	75.87 ± 2.62	74.06 ± 2.57	75.87 ± 1.88	75.87 ± 2.49	76.08	0.89
Type5	76.17 ± 1.18	76.61 ± 1.38	76.77 ± 1.82	77.23 ± 1.72	75.87 ± 4.11	78.13 ± 0.69	76.92 ± 0.91	77.68 ± 1.45	75.11 ± 0.79	75.42 ± 3.86	76.59	0.96
Type6	75.57 ± 1.63	75.27 ± 0.94	76.02 ± 1.97	74.66 ± 1.57	76.02 ± 1.97	76.32 ± 0.94	74.21 ± 1.21	73.61 ± 2.73	74.21 ± 2.07	75.87 ± 1.72	75.18	0.94
Type7	74.81 ± 2.65	76.69 ± 1.95	76.77 ± 0.26	76.47 ± 1.97	77.98 ± 0.69	75.27 ± 1.38	77.08 ± 0.94	74.96 ± 1.71	74.66 ± 2.39	75.57 ± 2.72	76.02	1.12
Type8	76.92 ± 1.57	74.78 ± 2.77	73.76 ± 1.81	75.56 ± 2.76	74.06 ± 3.85	73.91 ± 4.18	72.41 ± 1.63	73.15 ± 0.69	74.06 ± 2.57	72.55 ± 1.31	74.11	1.37
Type9	77.38 ± 0.46	76.08 ± 1.29	75.87 ± 1.83	77.22 ± 1.89	74.96 ± 3.66	74.66 ± 2.07	76.02 ± 2.52	76.17 ± 1.38	76.32 ± 2.32	76.74 ± 0.79	76.14	0.87
Average	76.49	76.23	75.99	76.47	76.09	75.80	75.85	75.58	75.52	75.85	75.99	0.24
	Robustness											
Type1	75.28 ± 1.22	74.45 ± 0.27	74.48 ± 1.82	74.51 ± 0.65	74.21 ± 1.25	74.09 ± 1.25	74.11 ± 2.33	74.37 ± 2.11	74.06 ± 2.12	74.43 ± 0.45	74.40	0.35
Type2	75.47 ± 0.34	74.66 ± 0.63	74.61 ± 0.23	74.65 ± 1.51	74.19 ± 0.54	74.01 ± 2.86	74.23 ± 1.04	74.62 ± 1.45	74.21 ± 3.21	74.01 ± 1.34	74.47	0.44
Type3	75.99 ± 0.84	76.02 ± 0.63	75.78 ± 2.14	75.81 ± 1.56	75.33 ± 2.36	75.54 ± 1.22	75.76 ± 0.56	75.12 ± 0.64	75.32 ± 0.76	75.21 ± 1.45	75.59	0.33
Type4	74.55 ± 1.34	74.43 ± 1.24	74.87 ± 1.29	75.32 ± 1.56	75.53 ± 0.65	75.23 ± 2.17	75.07 ± 1.45	74.01 ± 1.52	74.86 ± 2.01	74.69 ± 3.21	74.86	0.46
Type5	75.02 ± 1.89	75.46 ± 0.65	75.65 ± 0.92	75.71 ± 0.52	75.01 ± 2.11	76.12 ± 1.65	75.87 ± 2.12	75.91 ± 2.14	74.85 ± 0.76	74.89 ± 2.43	75.45	0.47
Type6	73.69 ± 0.93	72.95 ± 1.26	72.99 ± 0.76	73.51 ± 2.37	73.86 ± 1.37	73.88 ± 0.96	73.02 ± 3.22	72.56 ± 1.86	72.64 ± 1.75	72.66 ± 1.43	73.18	0.51
Type7	74.81 ± 1.35	75.33 ± 0.31	75.37 ± 0.96	75.37 ± 2.56	76.01 ± 0.96	75.01 ± 2.11	75.78 ± 0.43	74.92 ± 3.72	74.21 ± 2.01	74.26 ± 0.34	75.11	0.59
Type8	73.98 ± 0.97	72.62 ± 1.23	72.58 ± 1.21	73.52 ± 0.86	73.26 ± 2.15	73.06 ± 2.19	72.21 ± 0.54	72.65 ± 3.87	72.83 ± 1.43	72.02 ± 2.31	72.87	0.60
Type9	75.51 ± 1.01	74.45 ± 0.38	74.42 ± 1.33	75.62 ± 1.09	73.66 ± 1.96	73.58 ± 1.27	73.87 ± 1.44	73.89 ± 0.49	73.91 ± 1.05	73.86 ± 2.14	74.28	0.74
Average	74.92	74.49	74.53	74.89	74.56	74.50	74.43	74.23	74.10	74.00	74.47	0.13

Table 11
Results (average ± standard deviation, %) of trained GraphSAGE models with *Manifold-Mixup* (dataset: Politifact) using hybrid pooling operators with different α settings. The best results are marked in red color.

	Test Accuracy											
Mixup Ratio Alpha	**0.05**	**0.10**	**0.15**	**0.20**	**0.25**	**0.30**	**0.35**	**0.40**	**0.45**	**0.50**	Average	Std
Type1	78.88 ± 1.14	78.89 ± 1.88	78.58 ± 0.26	77.22 ± 1.89	76.32 ± 0.68	78.28 ± 0.79	77.98 ± 1.59	77.83 ± 0.91	78.28 ± 1.18	78.43 ± 0.68	78.07	0.79
Type2	76.62 ± 1.16	77.61 ± 0.32	76.62 ± 0.94	77.38 ± 0.79	79.19 ± 0.91	76.92 ± 0.91	77.07 ± 1.14	77.38 ± 0.92	77.83 ± 1.19	77.07 ± 0.66	77.37	0.75
Type3	79.18 ± 0.79	78.73 ± 0.67	77.68 ± 0.93	79.03 ± 0.53	78.58 ± 2.04	78.58 ± 1.14	79.33 ± 1.05	77.83 ± 0.79	78.73 ± 1.63	75.87 ± 1.72	78.35	1.02
Type4	76.32 ± 0.69	77.23 ± 0.55	76.77 ± 1.46	76.02 ± 1.19	77.38 ± 2.27	75.72 ± 1.31	77.37 ± 1.97	76.17 ± 0.69	77.38 ± 0.46	76.02 ± 0.45	76.64	0.66
Type5	77.53 ± 1.31	76.89 ± 0.99	77.22 ± 3.01	76.77 ± 2.91	77.38 ± 2.08	74.51 ± 1.83	74.66 ± 4.15	77.23 ± 2.04	73.61 ± 2.91	75.57 ± 1.19	76.14	1.43
Type6	74.66 ± 2.07	76.57 ± 0.23	75.31 ± 2.21	76.77 ± 3.02	74.96 ± 2.49	74.66 ± 4.07	75.42 ± 2.09	74.81 ± 1.72	74.81 ± 0.94	74.51 ± 1.88	75.25	0.80
Type7	76.63 ± 1.38	76.93 ± 1.28	74.51 ± 1.38	77.83 ± 0.45	76.93 ± 1.19	75.26 ± 1.14	76.77 ± 0.94	76.77 ± 0.94	75.87 ± 0.69	76.62 ± 1.83	76.41	0.95
Type8	76.47 ± 1.36	76.71 ± 0.97	75.41 ± 1.31	75.26 ± 2.32	74.96 ± 3.21	76.32 ± 1.83	75.26 ± 1.14	74.36 ± 2.76	76.77 ± 2.32	74.81 ± 1.38	75.63	0.86
Type9	75.41 ± 0.53	76.69 ± 0.32	75.11 ± 1.63	75.87 ± 1.14	74.51 ± 1.14	76.62 ± 1.86	72.71 ± 1.16	75.72 ± 0.69	74.06 ± 2.94	74.06 ± 1.45	75.08	1.25
Average	76.86	77.36	76.36	76.91	76.69	76.32	76.29	76.46	76.37	75.88	76.55	0.25
	Robustness											
Type1	77.26 ± 0.34	77.37 ± 0.63	77.12 ± 1.23	76.78 ± 2.31	76.01 ± 1.23	76.67 ± 1.34	76.56 ± 2.31	76.45 ± 1.45	76.54 ± 2.12	76.53 ± 1.23	76.73	0.42
Type2	76.25 ± 2.34	76.94 ± 1.19	76.23 ± 2.12	76.84 ± 1.23	77.21 ± 0.45	76.01 ± 1.36	76.34 ± 2.32	76.57 ± 2.56	76.87 ± 2.15	76.54 ± 2.45	76.58	0.38
Type3	75.96 ± 1.94	75.33 ± 2.24	74.84 ± 1.45	75.65 ± 1.12	75.12 ± 1.34	75.08 ± 3.21	75.35 ± 3.21	75.21 ± 3.45	75.32 ± 0.56	74.21 ± 2.19	75.21	0.47
Type4	75.54 ± 0.65	75.79 ± 0.33	75.34 ± 2.46	75.21 ± 0.53	75.87 ± 1.56	74.78 ± 3.56	75.98 ± 1.34	75.02 ± 2.11	75.09 ± 0.76	74.65 ± 1.94	75.33	0.46
Type5	75.89 ± 2.12	75.56 ± 1.27	75.65 ± 1.83	75.02 ± 1.34	75.98 ± 2.01	74.87 ± 0.45	74.45 ± 2.11	75.78 ± 0.56	75.02 ± 1.34	75.43 ± 1.45	75.37	0.50
Type6	73.21 ± 0.67	73.98 ± 1.59	73.01 ± 2.76	74.21 ± 2.94	73.03 ± 1.34	73.01 ± 2.12	73.88 ± 0.45	73.23 ± 0.54	73.12 ± 0.45	73.01 ± 0.54	73.37	0.47
Type7	71.71 ± 2.54	71.72 ± 0.32	71.12 ± 0.65	72.65 ± 2.61	72.45 ± 0.34	72.01 ± 0.45	71.98 ± 2.12	71.78 ± 1.23	71.45 ± 2.56	71.98 ± 0.76	71.89	0.44
Type8	74.02 ± 2.12	74.21 ± 1.17	73.87 ± 0.75	73.68 ± 3.21	73.54 ± 2.14	74.32 ± 0.46	74.12 ± 2.14	73.87 ± 1.65	73.92 ± 2.54	72.99 ± 0.87	73.85	0.38
Type9	75.12 ± 1.32	75.56 ± 0.66	75.02 ± 1.45	75.21 ± 0.87	75.01 ± 0.45	76.02 ± 1.23	71.98 ± 1.09	72.56 ± 0.87	71.88 ± 1.07	71.78 ± 1.47	74.01	1.73
Average	75.00	75.16	74.69	75.03	74.91	74.75	74.52	74.50	74.36	74.12	74.70	0.43

Table 9, Table 10, Table 11, and Table 12 present the detailed results of GNN models including GCN, GAT, GraphSAGE, and GIN on the Politifact dataset when using the Mixup training strategy with different α settings. First, we analyze how α affects each type of graph pooling method. Column *Std* shows the standard deviation of the results of each pooling method. We can see that, in the type level, α also

Table 12
Results (average ± standard deviation, %) of trained GIN models with *Manifold-Mixup* (dataset: Politifact) using hybrid pooling operators with different α settings. The best results are marked in red color.

	Test Accuracy											
Mixup Ratio Alpha	**0.05**	**0.10**	**0.15**	**0.20**	**0.25**	**0.30**	**0.35**	**0.40**	**0.45**	**0.50**	Average	Std
Type1	75.87 ± 2.14	78.28 ± 0.93	75.72 ± 1.83	74.96 ± 1.14	78.43 ± 1.45	77.98 ± 0.26	77.52 ± 0.46	76.77 ± 0.69	78.13 ± 1.59	76.47 ± 0.45	77.01	1.23
Type2	78.28 ± 0.91	78.06 ± 1.61	78.73 ± 1.26	78.28 ± 0.78	76.77 ± 0.69	78.28 ± 1.56	78.43 ± 2.14	78.13 ± 1.14	77.53 ± 0.26	77.38 ± 1.63	77.99	0.59
Type3	76.47 ± 0.45	77.92 ± 2.85	76.37 ± 2.95	75.57 ± 2.75	76.32 ± 2.72	75.72 ± 0.26	74.95 ± 1.13	76.62 ± 0.94	76.69 ± 0.32	76.62 ± 0.94	76.33	0.80
Type4	79.34 ± 0.26	77.74 ± 0.74	78.58 ± 2.23	77.83 ± 0.45	77.08 ± 1.38	77.08 ± 1.38	76.47 ± 0.78	77.98 ± 0.26	77.68 ± 0.25	77.58 ± 0.69	77.74	0.81
Type5	78.28 ± 0.92	77.64 ± 1.72	76.92 ± 0.45	78.43 ± 0.26	77.53 ± 1.71	78.43 ± 1.14	77.68 ± 0.52	77.98 ± 0.68	78.43 ± 1.14	77.68 ± 1.88	77.90	0.50
Type6	77.68 ± 1.14	78.05 ± 1.08	77.98 ± 1.38	78.88 ± 1.05	76.47 ± 1.63	77.23 ± 1.05	77.37 ± 0.79	77.53 ± 1.14	77.23 ± 1.45	77.98 ± 1.71	77.64	0.64
Type7	78.59 ± 1.05	77.83 ± 1.61	79.64 ± 0.45	79.34 ± 0.26	79.34 ± 0.69	79.04 ± 1.14	78.58 ± 0.26	77.83 ± 2.39	79.04 ± 0.69	79.94 ± 1.31	78.92	0.71
Type8	78.58 ± 0.26	78.39 ± 0.78	76.77 ± 2.61	78.43 ± 3.27	77.68 ± 1.83	78.28 ± 1.21	77.38 ± 1.63	78.13 ± 1.31	77.11 ± 1.86	76.98 ± 1.26	77.77	0.67
Type9	79.19 ± 0.91	77.74 ± 1.71	79.49 ± 1.14	79.49 ± 1.14	78.73 ± 0.46	78.73 ± 0.46	79.33 ± 1.75	78.58 ± 1.59	76.47 ± 1.97	76.92 ± 1.71	78.47	1.08
Average	78.03	**77.96**	77.80	77.91	77.59	77.86	77.52	77.73	77.59	77.51	77.75	0.23
	Robustness											
Type1	73.22 ± 1.34	73.31 ± 2.56	72.68 ± 2.34	71.87 ± 3.21	73.66 ± 2.34	73.01 ± 1.23	72.67 ± 1.23	72.21 ± 1.34	72.98 ± 2.23	71.67 ± 2.21	72.73	0.64
Type2	74.42 ± 2.45	74.21 ± 0.64	74.76 ± 0.45	74.54 ± 1.34	73.03 ± 2.12	74.21 ± 0.34	74.31 ± 1.45	74.23 ± 2.34	73.89 ± 0.45	73.55 ± 1.23	74.12	0.51
Type3	75.01 ± 3.21	75.06 ± 3.78	75.01 ± 0.65	74.76 ± 1.87	75.04 ± 1.34	74.87 ± 0.87	74.01 ± 0.54	74.87 ± 1.34	74.88 ± 1.23	74.76 ± 0.34	74.83	0.31
Type4	76.89 ± 0.65	75.56 ± 1.92	76.03 ± 1.45	75.87 ± 1.34	75.32 ± 2.32	75.21 ± 1.98	74.54 ± 1.45	75.02 ± 2.34	74.99 ± 2.12	74.77 ± 0.56	75.42	0.70
Type5	77.23 ± 1.45	76.41 ± 0.36	75.97 ± 3.21	76.76 ± 2.31	76.03 ± 2.98	76.76 ± 2.76	76.01 ± 2.12	76.32 ± 3.21	76.97 ± 1.23	76.74 ± 0.45	76.52	0.44
Type6	74.01 ± 0.45	74.66 ± 0.79	74.02 ± 0.45	74.98 ± 2.91	73.54 ± 1.65	73.98 ± 1.45	73.99 ± 2.43	74.04 ± 0.98	74.01 ± 2.14	74.04 ± 2.12	74.13	0.40
Type7	75.98 ± 0.65	75.11 ± 0.66	76.67 ± 2.12	76.45 ± 1.45	76.37 ± 0.98	76.12 ± 0.76	76.01 ± 2.54	75.67 ± 0.56	75.96 ± 1.45	75.99 ± 1.22	76.03	0.43
Type8	57.65 ± 1.34	57.19 ± 2.36	56.57 ± 0.54	57.03 ± 2.45	56.94 ± 1.23	57.01 ± 0.34	56.34 ± 0.65	56.54 ± 1.03	55.67 ± 2.34	55.23 ± 0.46	56.62	0.72
Type9	75.65 ± 2.34	74.36 ± 2.28	75.21 ± 2.01	75.19 ± 2.11	74.89 ± 2.41	74.78 ± 1.02	75.21 ± 0.54	74.98 ± 1.12	73.78 ± 0.45	73.84 ± 0.66	74.79	0.62
Average	73.34	72.87	72.99	73.05	72.76	72.88	72.57	72.65	72.57	72.29	72.80	0.15

Table 13
Results of statistical testing on Accuracy & Robustness. True: indicates that the comparison is statistically significant after the *Bonferroni correction* adjustment. Statistical testing methods: *Wilcoxon signed-rank test*. A gray background highlights the result marked as True.

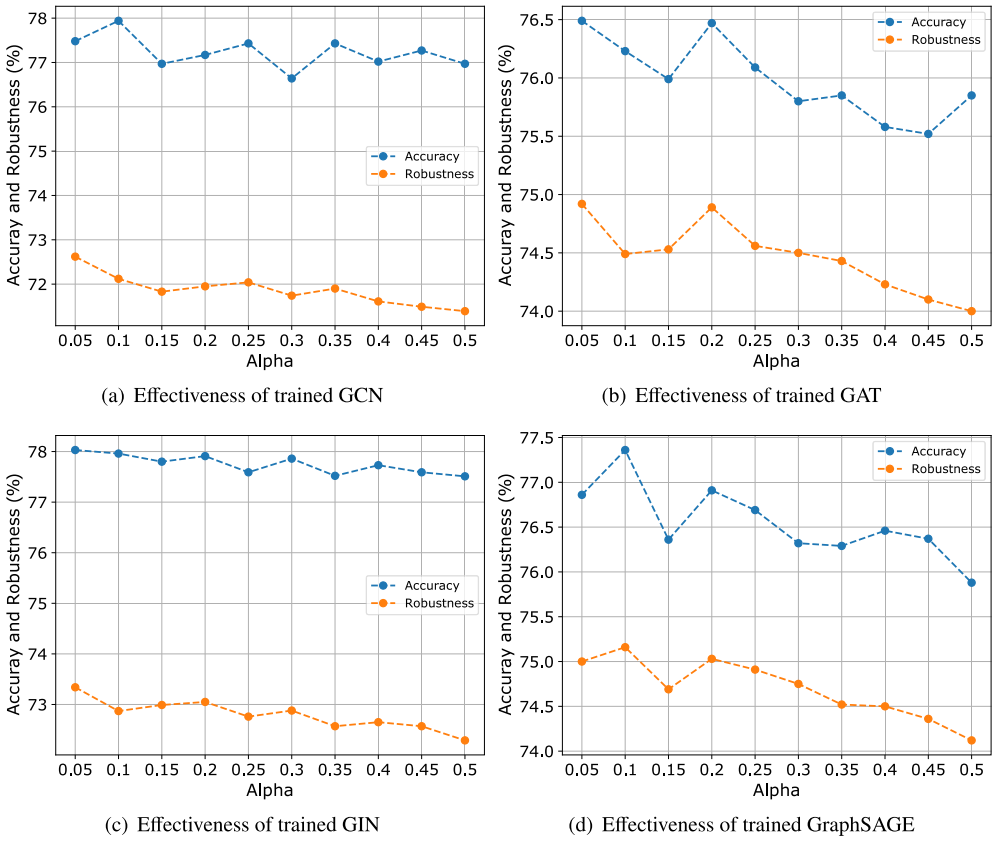
		Test Accuracy							
	Mixup Ratio Alpha	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
GCN	0.05	–	–	–	–	–	–	–	–
	0.10	True	True	–	–	–	–	–	–
GAT	0.05	–	–	–	–	–	–	True	–
	0.10	–	–	–	–	–	–	True	–
GraphSAGE	0.05	–	–	–	–	–	–	–	True
	0.10	True	–	–	True	True	True	True	True
GIN	0.05	–	–	–	–	–	–	–	–
	0.10	–	–	–	–	–	–	–	–
		Robustness							
GCN	0.05	True	–	–	True	–	True	True	True
	0.10	–	–	–	–	–	–	–	–
GAT	0.05	–	–	–	–	–	True	True	True
	0.10	–	True	–	–	–	True	True	True
GraphSAGE	0.05	True	–	–	–	–	True	True	True
	0.10	True	–	–	True	–	True	True	True
GIN	0.05	–	–	True	–	–	True	–	True
	0.10	–	–	–	–	–	–	–	–

does not greatly impact the Mixup training. The maximum standard deviations are only 1.84% (Accuracy of Type 3, Table 9) and 1.73% (Robustness of Type 9, Table 11). Maybe interestingly, we can find the two most stable and least stable types, i.e., The operator Type 4 ($\mathcal{M}_{sum}(\mathcal{P}_{att}, \mathcal{P}_{sum})$) from Table 9 and Type 2 ($\mathcal{M}_{mul}(\mathcal{P}_{att}, \mathcal{P}_{max})$) from Table 11 always have the smallest standard deviation values for both accuracy and robustness. In the Average row, across all hybrid pooling operators, the greatest average accuracy improvement is observed at α values of 0.05 and 0.10, analyzed across all GNNs. When considering robustness, except for GraphSAGE, which demonstrates the best average robustness improvement when α equals 0.10, the other GNNs show that setting α to 0.05 yields the best average robustness improvement compared to other α value settings.

Fig. 4 depicts the trending of average (e.g., row *Average* in Table 9) results of all GNN models. Then, we can see that although both the accuracy and robustness difference between using different α is not that big, i.e., the gap between accuracy (robustness) is 1.30% (1.23%). The smaller α can produce models with better performance, which is consistent with the conclusion of the original Mixup work (Zhang et al., 2018a). Especially for the robustness, there is a clear decreasing trend when α becomes bigger.

Statistical analysis. We conduct statistical testing to evaluate the significance of both accuracy and robustness improvement achieved by using hybrid operators at different Mixup ratios α. As depicted in Table 13, overall, the comparison becomes more statistically significant as the value of α increases, in terms of accuracy and robustness. However, there are a few cases, such as Test Accuracy-GIN, where the comparisons are consistently not statistically significant regardless of the increase in the value of α. We recommend setting $\alpha = 0.10$ for GNNs training to improve accuracy, as in 2 out of 3 cases, it has shown to be statistically significant compared to others. Moving to robustness, we recommend setting $\alpha = 0.05$ for GNN training, as it consistently shows statistical significance compared to other α settings across all GNN models. Additionally, $\alpha = 0.10$ is statistically significant compared to other α settings only in GraphSAGE and GAT.

In conclusion, *Manifold-Mixup* for graph-structured data with hybrid graph pooling is resilient to the α setting. Moreover, small α is recommended in the practical usage of Mixup. Although the operator Type



(a) Effectiveness of trained GCN

(b) Effectiveness of trained GAT

(c) Effectiveness of trained GIN

(d) Effectiveness of trained GraphSAGE

Fig. 4. Accuracy and Robustness trendings with different α.

1 is the best for producing high-performance GNN models, Type 4 for GCN and Type 2 for GraphSAGE are the most stable methods that are least affected by α.

Answer to RQ3: The performance (both clean accuracy and robustness) of GNN models is not sensitive (with a less than 2% difference) to the change of parameter α. A small α (e.g., 0.05 and 0.10) is recommended for GNNs training with *Manifold-Mixup*.

6. Threats to validity

The internal threat to validity comes from implementing the GNNs, each pooling method, and the Mixup for graph-structured data. The implementation of GNNs is based on Fey and Lenssen (2019), and the implementation of Mixup for graph data is based on the officially released project of Mixup (Zhang et al., 2018a).

External validity threats lie in the selected NLP and PL tasks, datasets, and GNNs. We consider both the traditional NLP tasks (text level) and PL tasks (source code level) in the study and include two datasets for each task. Particularly, for the NLP task, we consider two well-studied datasets and four types of node embedding. For the PL task, we include two popular programming languages (Java and Python). For the GNN models, we consider six famous graph neural networks: GCN, GCN-Virtual, GIN, GIN-Virtual, GAT, and GraphSAGE.

The construct threats to validity mainly come from the parameters of Mixup, randomness, and evaluation measures. Mixup only contains the parameter α that controls the weight of mixing two input instances. We follow the recommendation of the original Mixup algorithm and investigate the impact of this parameter. Moreover, further, we explore the impact of α in our study. To reduce the impact of randomness, we repeat each experiment five times with different random seeds and report the average and standard deviation results. Finally, concerning the evaluation measures, we consider both the test accuracy of the original test data and the robustness of corrupted test data. The latter one is specific for evaluating the generalization ability of GNNs.

7. Conclusions

In this paper, we comprehensively investigated how the graph pooling methods impact the effectiveness of Mixups when dealing with graph-structured data. We considered the Max-pooling operator, the state-of-the-art GMT pooling operator, and nine different hybrid pooling methods defined ourselves. In the empirical analysis part, we conducted experiments on both traditional NLP tasks (fake news detection) and PL tasks (problem classification) using six types of GNN architecture. The experimental results demonstrated that the pooling operator significantly impacts the effectiveness of Mixup, where hybrid pooling operators outperform both the Max-pooling and GMT operators in terms of producing accurate and robust GNN models. The hyperparameter α has a limited impact on Mixup in augmenting graph-structured data. This study gave the lesson that, when using Mixup in GNNs, carefully choosing the pooling operators could help produce better models.

CRediT authorship contribution statement

Zeming Dong: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Resources, Validation, Visualization, Writing – original draft, Writing – review & editing. **Qiang Hu:** Conceptualization, Writing – review & editing. **Zhenya Zhang:** Writing – review & editing. **Yuejun Guo:** Conceptualization. **Maxime Cordy:** Supervision. **Mike Papadakis:** Supervision. **Yves Le Traon:** Supervision. **Jianjun Zhao:** Funding acquisition, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

https://github.com/zemingd/HybridPool4Mixup.

Acknowledgment

This research is supported in part by JSPS KAKENHI Grant No. JP23H03372 and No. JP24K02920, Japan. The research is also supported in part by JST-Mirai Program Grant No. JPMJMI20B8.

References

Allamanis, M., Barr, E.T., Devanbu, P., Sutton, C., 2018a. A survey of machine learning for big code and naturalness. ACM Comput. Surv. 51 (4), 81.

Allamanis, M., Brockschmidt, M., Khademi, M., 2018b. Learning to represent programs with graphs. In: International Conference on Learning Representations. URL https://openreview.net/forum?id=BJOFETxR-.

Allamanis, M., Jackson-Flux, H., Brockschmidt, M., 2021. Self-supervised bug detection and repair. Adv. Neural Inf. Process. Syst. 34, 27865–27876.

Armstrong, R.A., 2014. When to use the B onferroni correction. Ophthalmic Physiol. Opt. 34 (5), 502–508.

Atwood, J., Towsley, D., 2016. Diffusion-convolutional neural networks. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. NIPS '16, Curran Associates Inc., Red Hook, NY, USA, pp. 2001–2009.

Baek, J., Kang, M., Hwang, S.J., 2021. Accurate learning of graph representations with graph multiset pooling. In: International Conference on Learning Representations. URL https://openreview.net/forum?id=JHcqXGaqiGn.

Bahdanau, D., Cho, K., Bengio, Y., 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Bianchi, F.M., Grattarola, D., Alippi, C., 2020. Spectral clustering with graph neural networks for graph pooling. In: International Conference on Machine Learning. PMLR, pp. 874–883.

Cangea, C., Veličković, P., Jovanović, N., Kipf, T., Liò, P., 2018. Towards sparse hierarchical graph classifiers. arXiv preprint arXiv:1811.01287.

Chen, J., Tam, D., Raffel, C., Bansal, M., Yang, D., 2023. An empirical survey of data augmentation for limited data learning in NLP. Trans. Assoc. Comput. Linguist. 11, 191–211.

Chen, J., Yang, Z., Yang, D., 2020. MixText: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Online, pp. 2147–2157. http://dx.doi.org/10.18653/v1/2020.acl-main.194, URL https://aclanthology.org/2020.acl-main.194.

Dinella, E., Dai, H., Li, Z., Naik, M., Song, L., Wang, K., 2020. Hoppity: Learning graph transformations to detect and fix bugs in programs. In: International Conference on Learning Representations. ICLR.

Dong, Z., Hu, Q., Guo, Y., Cordy, M., Papadakis, M., Zhang, Z., Traon, Y.L., Zhao, J., 2023a. MixCode: Enhancing code classification by mixup-based data augmentation. In: 2023 IEEE International Conference on Software Analysis, Evolution and Reengineering. SANER, pp. 379–390. http://dx.doi.org/10.1109/SANER56733.2023.00043.

Dong, Z., Hu, Q., Guo, Y., Zhang, Z., Cordy, M., Papadakis, M., Traon, Y.L., Zhao, J., 2023b. Boosting source code learning with data augmentation: An empirical study. arXiv preprint arXiv:2303.06808.

Dong, Z., Hu, Q., Zhang, Z., Zhao, J., 2024. On the effectiveness of graph data augmentation for source code learning. Knowl.-Based Syst. 285, 111328. http://dx.doi.org/10.1016/j.knosys.2023.111328.

Dou, Y., Shu, K., Xia, C., Yu, P.S., Sun, L., 2021. User preference-aware fake news detection. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '21, Association for Computing Machinery, New York, NY, USA, pp. 2051–2055. http://dx.doi.org/10.1145/3404835.3462990.

Fabian, B., Baumann, A., Lackner, J., 2015. Topological analysis of cloud service connectivity. Comput. Ind. Eng. 88 (C), 151–165. http://dx.doi.org/10.1016/j.cie.2015.06.009.

Feng, W., Zhang, J., Dong, Y., Han, Y., Luan, H., Xu, Q., Yang, Q., Kharlamov, E., Tang, J., 2020. Graph random neural networks for semi-supervised learning on graphs. In: NeurIPS, vol. 33, pp. 22092–22103.

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

Gao, H., Ji, S., 2019. Graph U-Nets. In: International Conference on Machine Learning. PMLR, pp. 2083–2092.

Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E., 2017. Neural message passing for quantum chemistry. In: Precup, D., Teh, Y.W. (Eds.), Proceedings of the 34th International Conference on Machine Learning. In: Proceedings of Machine Learning Research, vol. 70, PMLR, pp. 1263–1272, URL https://proceedings.mlr.press/v70/gilmer17a.html.

Grattarola, D., Zambon, D., Bianchi, F.M., Alippi, C., 2022. Understanding pooling in graph neural networks. IEEE Trans. Neural Netw. Learn. Syst. 1–11. http://dx.doi.org/10.1109/TNNLS.2022.3190922.

Guo, H., Mao, Y., Zhang, R., 2019. Augmenting data with mixup for sentence classification: An empirical study. arXiv preprint arXiv:1905.08941.

Hamilton, W., Ying, Z., Leskovec, J., 2017. Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, vol. 30.

Huang, L., Ma, D., Li, S., Zhang, X., Wang, H., 2019. Text level graph neural network for text classification. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing. EMNLP-IJCNLP, Association for Computational Linguistics, Hong Kong, China, pp. 3444–3450. http://dx.doi.org/10.18653/v1/D19-1345, URL https://aclanthology.org/D19-1345.

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kipf, T.N., Welling, M., 2017. Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations. URL https://openreview.net/forum?id=SJU4ayYgl.

Knyazev, B., Taylor, G.W., Amer, M.R., 2019. Understanding attention and generalization in graph neural networks. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA.

Konno, R., Matsubayashi, Y., Kiyono, S., Ouchi, H., Takahashi, R., Inui, K., 2020. An empirical study of contextual data augmentation for Japanese zero anaphora resolution. In: Proceedings of the 28th International Conference on Computational Linguistics. International Committee on Computational Linguistics, Barcelona, Spain (Online), pp. 4956–4968. http://dx.doi.org/10.18653/v1/2020.coling-main.435, URL https://aclanthology.org/2020.coling-main.435.

Lee, J., Lee, I., Kang, J., 2019. Self-attention graph pooling. In: International Conference on Machine Learning. PMLR, pp. 3734–3743.

Li, J., Rong, Y., Cheng, H., Meng, H., Huang, W., Huang, J., 2019. Semi-supervised graph classification: A hierarchical graph perspective. In: The World Wide Web Conference. pp. 972–982.

Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.S., 2016. Gated graph sequence neural networks. In: Bengio, Y., LeCun, Y. (Eds.), 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings. URL http://arxiv.org/abs/1511.05493.

Ma, Y., Wang, S., Aggarwal, C.C., Tang, J., 2019. Graph convolutional networks with eigenpooling. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 723–731.

Mesquita, D., Souza, A.H., Kaski, S., 2020. Rethinking pooling in graph neural networks. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20, Curran Associates Inc., Red Hook, NY, USA.

Neyshabur, B., Bhojanapalli, S., McAllester, D., Srebro, N., 2017. Exploring generalization in deep learning. In: Advances in Neural Information Processing Systems, vol. 30.

Nguyen, V.-A., Nguyen, V., Le, T., Tran, Q.H., Phung, D., et al., 2022. ReGVD: Revisiting graph neural networks for vulnerability detection. In: 2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings. ICSE-Companion, IEEE, pp. 178–182.

Oehlers, M., Fabian, B., 2021. Graph metrics for network robustness—A survey. Mathematics 9 (8), http://dx.doi.org/10.3390/math9080895, URL https://www.mdpi.com/2227-7390/9/8/895.

Papp, P.A., Martinkus, K., Faber, L., Wattenhofer, R., 2021. DropGNN: Random dropouts increase the expressiveness of graph neural networks. Adv. Neural Inf. Process. Syst. 34, 21997–22009.

Puri, R., Kung, D.S., Janssen, G., Zhang, W., Domeniconi, G., Zolotov, V., Dolby, J., Chen, J., Choudhury, M., Decker, L., et al., 2021. CodeNet: A large-scale AI for code dataset for learning a diversity of coding tasks. arXiv preprint arXiv:2105.12655.

Ranjan, E., Sanyal, S., Talukdar, P., 2020. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, (no. 04), pp. 5470–5477.

Rong, Y., Huang, W., Xu, T., Huang, J., 2020. DropEdge: towards deep graph convolutional networks on node classification. In: ICLR.

Shorten, C., Khoshgoftaar, T.M., 2019. A survey on image data augmentation for deep learning. J. Big Data 6 (1), 1–48.

Simonovsky, M., Komodakis, N., 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3693–3702.

Sun, L., Xia, C., Yin, W., Liang, T., Yu, P., He, L., 2020. Mixup-transformer: Dynamic data augmentation for NLP tasks. In: Proceedings of the 28th International Conference on Computational Linguistics. International Committee on Computational Linguistics, Barcelona, Spain (Online), pp. 3436–3440. http://dx.doi.org/10.18653/v1/2020.coling-main.305, URL https://aclanthology.org/2020.coling-main.305.

Tu, Y., 2000. How robust is the internet? Nature 406 (6794), 353–354.
Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., 2017. Graph attention networks. stat 1050, 20.
Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., Bengio, Y., 2019. Manifold mixup: Better representations by interpolating hidden states. In: Chaudhuri, K., Salakhutdinov, R. (Eds.), Proceedings of the 36th International Conference on Machine Learning. In: Proceedings of Machine Learning Research, vol. 97, PMLR, pp. 6438–6447, URL https://proceedings.mlr.press/v97/verma19a.html.
Wang, Y.G., Li, M., Ma, Z., Montúfar, G., Zhuang, X., Fan, Y., 2019. Haarpooling: Graph pooling with compressive haar basis.
Wang, W., Li, G., Ma, B., Xia, X., Jin, Z., 2020a. Detecting code clones with graph neural network and flow-augmented abstract syntax tree. In: 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering. SANER, pp. 261–271. http://dx.doi.org/10.1109/SANER48275.2020.9054857.
Wang, Y., Wang, W., Liang, Y., Cai, Y., Hooi, B., 2020b. GraphCrop: Subgraph cropping for graph classification. arXiv abs/2009.10564, URL https://api.semanticscholar.org/CorpusID:221836528.
Wang, Y., Wang, W., Liang, Y., Cai, Y., Hooi, B., 2021. Mixup for node and graph classification. In: Proceedings of the Web Conference 2021. pp. 3663–3674.
Woolson, R.F., 2007. Wilcoxon signed-rank test. In: Wiley Encyclopedia of Clinical Trials. Wiley Online Library, pp. 1–3.
Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S., 2021. A comprehensive survey on graph neural networks. IEEE Trans. Neural Netw. Learn. Syst. 32 (1), 4–24. http://dx.doi.org/10.1109/TNNLS.2020.2978386.
Xu, K., Hu, W., Leskovec, J., Jegelka, S., 2019. How powerful are graph neural networks? In: International Conference on Learning Representations. URL https://openreview.net/forum?id=ryGs6iA5Km.
Xu, H., Mannor, S., 2012. Robustness and generalization. Mach. Learn. 86 (3), 391–423. http://dx.doi.org/10.1007/s10994-011-5268-1.
Yao, L., Mao, C., Luo, Y., 2019. Graph convolutional networks for text classification. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, (no. 01), pp. 7370–7377.
Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J., 2018. Hierarchical graph representation learning with differentiable pooling. In: Advances in Neural Information Processing Systems, vol. 31.
Yoon, S., Kim, G., Park, K., 2021. SSMix: Saliency-based span mixup for text classification. In: Findings of the Association for Computational Linguistics. ACL-IJCNLP 2021, Association for Computational Linguistics, Online, pp. 3225–3234. http://dx.doi.org/10.18653/v1/2021.findings-acl.285, URL https://aclanthology.org/2021.findings-acl.285.
Yu, S., Wang, T., Wang, J., 2022. Data augmentation by program transformation. J. Syst. Softw. 190, 111304. http://dx.doi.org/10.1016/j.jss.2022.111304, URL https://www.sciencedirect.com/science/article/pii/S0164121222000541.
Yuan, H., Ji, S., 2020. Structpool: Structured graph pooling via conditional random fields. In: Proceedings of the 8th International Conference on Learning Representations.
Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y., 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6023–6032.
Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D., 2018a. mixup: Beyond empirical risk minimization. In: International Conference on Learning Representations. URL https://openreview.net/forum?id=r1Ddp1-Rb.
Zhang, M., Cui, Z., Neumann, M., Chen, Y., 2018b. An end-to-end deep learning architecture for graph classification. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, (no. 1).
Zhang, L., Deng, Z., Kawaguchi, K., Ghorbani, A., Zou, J., 2021. How does mixup help with robustness and generalization? In: International Conference on Learning Representations. URL https://openreview.net/forum?id=8yKEo06dKNo.
Zhang, L., Deng, Z., Kawaguchi, K., Zou, J., 2022. When and how mixup improves calibration. In: Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S. (Eds.), Proceedings of the 39th International Conference on Machine Learning. In: Proceedings of Machine Learning Research, vol. 162, PMLR, pp. 26135–26160, URL https://proceedings.mlr.press/v162/zhang22f.html.
Zhang, Y., Yu, X., Cui, Z., Wu, S., Wen, Z., Wang, L., 2020a. Every document owns its structure: Inductive text classification via graph neural networks. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Online, pp. 334–339. http://dx.doi.org/10.18653/v1/2020.acl-main.31, URL https://aclanthology.org/2020.acl-main.31.
Zhang, R., Yu, Y., Zhang, C., 2020b. SeqMix: Augmenting active sequence labeling via sequence mixup. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. EMNLP, Association for Computational Linguistics, Online, pp. 8566–8579. http://dx.doi.org/10.18653/v1/2020.emnlp-main.691, URL https://aclanthology.org/2020.emnlp-main.691.
Zhao, T., Jin, W., Liu, Y., Wang, Y., Liu, G., Günneman, S., Shah, N., Jiang, M., 2023. Graph data augmentation for graph machine learning: A survey. IEEE Data Eng. Bull..
Zhou, Y., Liu, S., Siow, J., Du, X., Liu, Y., 2019. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. Curran Associates Inc, Red Hook, NY, USA.

Zeming Dong received his Bachelor's degree at Yancheng Institute of Technology, China, and his Master of Engineering degree in Electrical and Electronic Information Engineering at Yamaguchi University, Japan, respectively. He is now pursuing a Ph.D. degree at Kyushu University, Japan. His research interests include software engineering and machine learning.

Qiang Hu got his Ph.D. degree from the University of Luxembourg. He got his Master's degree from Kyushu University and B.S. degree from the University of Electronic Science and Technology of China.

Zhenya Zhang is an assistant professor in Faculty of Information Science and Electrical Engineering, Kyushu University, Japan. He obtained his Ph.D. from the National Institute of Informatics, the Graduate University for Advanced Studies (SOKENDAI), Japan, in 2020. His research interests include quality assurance of software systems via formal methods.

Yuejun Guo got the Ph.D. from the UdG in May 2020. She got the master degree in computer science and technology from Tianjin University in 2016 and the bachelor degree in computer science and technology from Civil Aviation University of China in 2013. Her main research interests focus on trajectory data analysis including clustering, anomaly detection and map construction. She also worked on image processing.

Maxime Cordy received his Ph.D. degree in 2014 from the University of Namur, Belgium. He has worked as a Research Scientist with the Interdisciplinary Center for Security, Reliability and Trust (SnT) with the University of Luxembourg. His research fields include software engineering, software quality assurance, model checking, and applied artificial intelligence.

Mike Papadakis received the Ph.D. diploma degree in computer science from the Athens University of Economics and Business. He is an associate professor with the Interdisciplinary Center for Security, Reliability and Trust (SnT) with the University of Luxembourg. He is recognized for his work on software testing and in particular in the area of mutation testing. His research interests also include static analysis, prediction modeling and search-based software engineering.

Yves Le Traon is a professor with the University of Luxembourg where he leads the SERVAL (SEcurity, Reasoning and VALidation) research team. His research interests within the group include (1) innovative testing and debugging techniques, (2) Android apps security and reliability using static code analysis, machine learning techniques and, model driven engineering with a focus on IoT and CPS. His reputation in the domain of software testing is acknowledged by the community. He has been General Chair of major conferences in the domain, such as the 2013 IEEE International Conference on Software Testing, Verification and Validation (ICST), and Program Chair of the 2016 IEEE International Conference on Software Quality, Reliability and Security (QRS). He serves at the editorial boards of several, internationally-known journals (STVR, SoSym, IEEE Transactions on Reliability) and is author of more than 150 publications in international peer-reviewed conferences and journals.

Jianjun Zhao is a professor at the School of Information Science and Electrical Engineering at Kyushu University. He received his B.S. degree from Tsinghua University, China, in 1987 and his Ph.D. from Kyushu University, Japan, in 1997, both in computer science. He was an assistant/associate professor at Fukuoka Institute of Technology, Japan (1997–2005), and a professor at Shanghai Jiao Tong University, China (2005–2016). He was a visiting scientist at MIT Computer Science and Artificial Intelligence Laboratory from 2002–2003. His research interests include software engineering and programming languages for classical and non-classical computing.