Software Engineering Group
Department of Computer Science
Nanjing University
http://seg.nju.edu.cn

Technical Report No. NJU-SEG-2025-IJ-003

2025-IJ-003

Improving Graph Learning-Based Fault Localization with Tailored Semi-supervised Learning

CHUNLI，HUILI，ZHONGLI，MINXUEPAN，XUANDONGLI

Technical Report 2025

Improving Graph Learning-Based Fault Localization with Tailored Semi-supervised Learning

CHUN LI, Nanjing University, China
HUI LI, Samsung Electronics (China) R&D Centre, China
ZHONG LI, Nanjing University, China
MINXUE PAN[*], Nanjing University, China
XUANDONG LI, Nanjing University, China

Due to advancements in graph neural networks, graph learning-based fault localization (GBFL) methods have achieved promising results. However, as these methods are supervised learning paradigms and deep learning is typically data-hungry, they can only be trained on fully labeled large-scale datasets. This is impractical because labeling failed tests is similar to manual fault localization, which is time-consuming and labor-intensive, leading to only a small portion of failed tests that can be labeled within limited budgets. These data labeling limitations would lead to the sub-optimal effectiveness of supervised GBFL techniques. Semi-supervised learning (SSL) provides an effective means of leveraging unlabeled data to improve a model's performance and address data labeling limitations. However, as these methods are not specifically designed for fault localization, directly utilizing them might lead to sub-optimal effectiveness. In response, we propose a novel semi-supervised GBFL framework, LEGATO. LEGATO first leverages the attention mechanism to identify and augment likely fault-unrelated sub-graphs in unlabeled graphs and then quantifies the suspiciousness distribution of unlabeled graphs to estimate pseudo-labels. Through training the model on augmented unlabeled graphs and pseudo-labels, LEGATO can utilize the unlabeled data to improve the effectiveness of fault localization and address the restrictions in data labeling. By extensive evaluations against 3 baselines SSL methods, LEGATO demonstrates superior performance by outperforming all the methods in comparison.

CCS Concepts: • **Software and its engineering** → **Software testing and debugging**.

Additional Key Words and Phrases: Fault localization, semi-supervised learning, graph learning.

ACM Reference Format:
Chun Li, Hui Li, Zhong Li, Minxue Pan, and Xuandong Li. 2025. Improving Graph Learning-Based Fault Localization with Tailored Semi-supervised Learning. *Proc. ACM Softw. Eng.* 2, FSE, Article FSE069 (July 2025), 23 pages. https://doi.org/10.1145/3715788

1 Introduction

Locating software faults is the critical first step in the debugging process. Manual fault localization is often a time-consuming and labor-intensive process [7, 37]. Hence, extensive research has been conducted on automated fault localization techniques [9, 20, 24, 28–30, 62, 63, 67] to fully automate the process of diagnose fault program entities (i.e., files or methods) and facilitate the software

[*] Corresponding author.

Authors' Contact Information: Chun Li, Nanjing University, State Key Laboratory for Novel Software Technology, Nanjing, China, chunli@smail.nju.edu.cn; Hui Li, Samsung Electronics (China) R&D Centre, Nanjing, China, hui.li@samsung.com; Zhong Li, Nanjing University, State Key Laboratory for Novel Software Technology, Nanjing, China, lizhong@nju.edu.cn; Minxue Pan, Nanjing University, State Key Laboratory for Novel Software Technology, Nanjing, China, mxp@nju.edu.cn; Xuandong Li, Nanjing University, State Key Laboratory for Novel Software Technology, Nanjing, China, lxd@nju.edu.cn.

ACM 2994-970X/2025/7-ARTFSE069
https://doi.org/10.1145/3715788

debugging process. To date, researchers have proposed various fault localization techniques, and learning-based fault localization (LBFL) has been intensively studied in the literature due to its effectiveness and recent advances in machine/deep learning [20, 24, 26, 28, 30, 31, 40, 65, 67, 69, 76]. LBFL leverages learning techniques to train models to calculate the suspiciousness scores of program entities and rank them.

In LBFL techniques, methods based on graph neural networks (GNNs) have evolved significantly and shown promising results [22, 30, 38–40, 59, 64]. Graph learning-based fault localization (GBFL) represents the program semantics during the testing by graphs. For example, GRACE [30] uses the nodes in the abstract syntax tree (AST) of the source code and test cases as nodes, and represents the coverage information as edges between test nodes and statement nodes to construct graphs. Current GBFL methods are typically based on the paradigm of supervised learning, which implies that existing methods require training on a fully labeled dataset. Given that deep networks are typically data-hungry [16, 47, 61], using GBFL methods necessitates significant time and cost to label sufficient data. However, this is impractical for two reasons. First, in fault localization, labeling the fault entities in failed tests is similar to manual fault localization. As mentioned before, manual fault localization is a time-consuming and labor-intensive process, making it challenging to label sufficient data. Second, software typically undergoes extensive testing before being released to users. Thus, a large number of unlabeled failed tests are generated. However, given limited budgets, engineers may only be able to label a small portion of them. These restrictions would lead to the sub-optimal effectiveness of supervised GBFL techniques in real-world contexts.

To tackle the restrictions in data labeling, researchers extend supervised learning to semi-supervised learning (SSL) to leverage unlabeled samples and enhance model training [21, 49]. Currently, cutting-edge research in SSL [6, 8, 47, 55] can be viewed as first producing a pseudo-label for unlabeled samples [17, 21], then training the model to predict the pseudo-label when fed augmented versions of the same input [4, 43]. However, these methods are challenging to adopt in graph learning-based fault localization. The main challenges stem from the following two aspects. **First**, since fault localization requires the model to identify suspicious program entities based on the fault context, preserving the fault context as much as possible during the augmentation is crucial. Existing graph augmentation techniques [75] primarily rely on the degree of nodes to calculate the importance of edges and drop those edges deemed less important. However, the degree of nodes cannot indicate its relevance to faults, leading to potential disruption of the fault context and sub-optimal effectiveness. **Second**, SSL methods typically rely on thresholds to select high-confidence outputs as pseudo-labels. However, it is non-trivial as a lower threshold might introduce incorrect or noisy labels, and a higher threshold might lead to overfitting [21]. Furthermore, the possibility of multiple fault entities in fault localization makes the selection more difficult. Existing dynamic threshold methods adjust thresholds across various classes by estimating the learning state of the model for each class [55, 68]. However, these methods are unsuitable for fault localization, as fault localization involves ranking rather than classification.

In this paper, we propose Legato (semi-supervised **LE**arnin**G**-b**A**sed faul**T** l**O**calization), a novel semi-supervised graph learning-based fault localization framework specifically designed for only a small portion of failed tests are labeled. Specifically, the key insight of Legato to address the aforementioned limitations is two-fold.

First, we propose to leverage the attention mechanism to guide us in identifying and augmenting fault-unrelated sub-graphs in unlabeled graphs. The intuition is that the attention mechanism reflects the degree of focus the model places on different nodes and edges during the decision-making process [32, 50]. Therefore, when we train an attention-based GNN to classify the failed graphs, the model's attention will be decreased on sub-graphs in the failed graphs that almost exclusively appear in passed graphs, which are more likely fault-unrelated. As such, based on the attention mechanism, we can identify sub-graphs likely to be fault-unrelated and augment them while minimizing the impact on fault context. Furthermore, training the model to generate consistent predictions before and after augmentation enables it to concentrate on invariant features likely related to faults, thereby enhancing fault localization.

Second, we introduce a threshold-free pseudo-label estimation method that quantifies the distribution of suspiciousness scores (i.e., confidence) output by the model for all fault entities. Then, if some program entities have significantly higher suspiciousness scores than others, we select them as pseudo-labels. The basic idea is that the suspiciousness scores of all target program entities form a distribution that sums to one. When the model assigns high suspiciousness scores to certain program entities, the scores for the remaining entities tend to be lower. Thus, by quantifying the distribution of suspiciousness scores across all target program entities, we can identify program entities with significantly high scores compared to others and select them as pseudo-labels without depending on specific thresholds or considering the number of faulty entities in unlabeled graphs.

Our evaluation of Legato's performance in a large-scale dataset contains 7,500 passed and 7,500 failed tests, of which only 8% failed tests have been labeled. This dataset comes from the testing of eleven software projects written in C/C++, each exceeding one million lines of code, supplied by our industrial partner which is a global corporation. We benchmarked Legato against three baseline SSL methods based on the same supervised GBFL, demonstrating its exceptional efficacy by significantly outperforming all compared methods, including a remarkable 18.05% and 38.82% method level improvement over the top-performing method in within-project and cross-project scenarios respectively.

The main contributions of this paper are as follows:

- We propose Legato, a novel semi-supervised graph learning-based fault localization framework, which trains the model both on unlabeled and labeled graphs to tackle the restrictions in data labeling.
- We develop attention-guided graph augmentation and pseudo-label estimation to train the model on the augmented version of unlabeled graphs and pseudo-labels to utilize the unlabeled graphs and improve the performance of fault localization.
- We conduct extensive evaluations of Legato within a semi-supervised context on a large-scale dataset, and the results demonstrate that Legato effectively utilizes unlabeled graphs for fault localization and outperforms baseline SSL methods in comparison.

2 Methodology

In this section, we first define our problem. Then, we discuss the primary challenges faced when utilizing SSL for fault localization tasks and introduce our idea of Legato to solve the aforementioned challenges.

2.1 Problem Statement

In this work, we target GBFL methods that utilize GNNs to learn program semantics and localize faults within software. Formally, let $D = \{(G_i, Y_i)_{i=1}^{n}\}$ denote the training dataset, where G_i is the graph constructed from a test suite contains at least one failed test case, and Y_i is the label matrix. Take the graph in GRACE [30] as an example, the nodes consist of the AST of the source code and test cases, while the edges are constructed based on coverage information. Let $\mathcal{V}$ denote the node set of G and the node $v_i \in \mathcal{V}$ represents the program entity involved in the test suite. If node v_i is a fault entity, the label $y_i \in Y$ of it is set to 1, and otherwise, y_i is set to 0. Then, based on the training dataset D, a GNN $\mathcal{F}$ is learned via minimizing the listwise loss function $\mathcal{L} = -\sum_{i=1}^{t} y_i \log(p(v_i))$, where t is the number of target nodes (such as method or statement nodes) in $\mathcal{V}$, and $p(v_i)$ is the suspiciousness score of the target node v_i output by the model. After training, the target nodes will be ranked based on their suspiciousness scores $p(v_i)$ output by the GNN $\mathcal{F}$.

As shown, the effectiveness of GBFL methods heavily depends on the quality of the training datasets D. However, labeling fault entities in failed tests is a challenging task that requires significant manual effort by experts, particularly when test volumes are high and resources are constrained [7, 40]. Consequently, the graphs in the training dataset D are generally only partially labeled in real-world scenarios [52, 58]. That is, the dataset D can be viewed as $D_l \cup D_u$, where $D_l = \{(G_i, Y_i)_{i=1}^{m}\}$ denotes the labeled dataset, $D_u = \{G_i\}_{i=1}^{k}$ denote the unlabeled dataset, and $|D_l| << |D_u|$. Training GNN models on such a dataset D would result in sub-optimal GBFL methods due to the limited supervision provided by the dataset [24, 47]. Thus, the problem we want to address in this paper is: *Given a dataset containing labeled and unlabeled graphs, how can we effectively and efficiently learn a fault localization model when only a small portion of graphs are labeled?*

Semi-supervised learning (SSL) [49] provides a promising solution for this problem. The key spirit of SSL is to generate pseudo-labels for the unlabeled samples and train the model to predict the pseudo-labels for augmented versions of unlabeled samples. Although the effectiveness of SSL techniques has been demonstrated in many domains, such as image recognition [6, 47, 68], natural language processing [13, 45], and object detection [33, 54], they still encounter several challenges when applying for training GBFL models.

- **Challenge I: How to perform graph augmentation on unlabeled graphs with less impact on fault context.** A key requirement of SSL is to generate effective augmentations for unlabeled samples. However, directly augmenting the graph (e.g., deleting several edges) without any guidance might disrupt the fault context, leading to noisy augmented views and sub-optimal effectiveness [70, 71, 75]. Therefore, it is critical to identify fault-unrelated sub-graphs and perform augmentations only on these, in order to minimize the impact on the fault context. Unfortunately, the information available from test results is insufficient for effectively detecting fault-unrelated sub-graphs. Specifically, all program entities involved in a failed test might be fault entities. Although passed tests provide some coverage information, coverage alone struggles to capture the complex program semantics in the graph, such as code syntax structure. Hence, an effective graph augmentation mechanism that can identify and perform augmentations on fault-unrelated sub-graphs is desirable for applying SSL to GBFL methods.
- **Challenge II: How to select optimal threshold for pseudo-labeling to unlabeled graphs.** Existing pseudo-labeling methods typically employ a threshold to determine the high-confidence outputs as pseudo-labels. However, setting an optimal threshold for pseudo-labeling in fault localization is challenging due to the uncertainty in the number of faulty entities in failed tests. When the threshold is too low, noisy labels may be introduced in unlabeled graphs with fewer faulty entities. Conversely, if the threshold is too high, some true faulty entities may be missed in unlabeled graphs, limiting the information that can be learned. Recently, several approaches [55, 68] have been developed to adaptively select thresholds for pseudo-labeling. However, most of these adaptive threshold methods are tailored for classification problems, which involve assessing the model's learning status for each class. This makes it challenging to apply adaptive threshold methods to GBFL methods because GBFL formulates fault localization as a ranking task. Transitively, we are required to evaluate the learning status of each program entity to set the adaptive thresholds, resulting in significant computational overhead. Therefore, it is important to develop a threshold-free pseudo-labeling approach to generate effective pseudo-labels when applying SSL to GBFL methods.

2.2 Our Approach

In this section, we introduce our idea to solve the aforementioned challenges.

Identifying likely fault-unrelated sub-graphs by attention mechanism and performing augmentation on them. To perform graph augmentation on the likely fault-unrelated sub-graphs, we propose to train the attention-based GNN to classify the passed and failed graphs and utilize the attention matrix to identify the likely fault-unrelated sub-graphs. The key insight is that, to achieve more effective classification, the attention mechanism is trained to capture distinct sub-graphs, and thus, the attention matrix can be used to evaluate the model's focus on different sub-graphs [32, 50, 51]. More specifically, when learning to classify failed graphs, the model will focus more on sub-graphs that appear only in failed graphs, which are more likely to be fault-related, and assign them higher attention weights. Conversely, the sub-graphs within the failed graphs that also frequently appear in passed graphs, which are more likely fault-unrelated, will receive lower attention. Therefore, we can input failed graphs into an attention-based GNN to evaluate the attention weights of different sub-graphs and identify those with the lowest weights as fault-unrelated sub-graphs. As such, by performing graph augmentation on these fault-unrelated sub-graphs, we can minimize the impact on the fault context. Furthermore, when we require the model to maintain consistent outputs before and after augmentation, the model will learn and focus more on the invariant features that are more likely fault-related, thereby enhancing fault localization by SSL. We will present details about how Legato identifies fault-unrelated sub-graphs and performs graph augmentation on them based on the attention mechanism in Section 3.2.

Estimating pseudo-labels by quantifying the distribution of suspiciousness scores among all program entities. For GBFL methods, we observe that the suspiciousness scores output by the models generally form a probability distribution that sums to one. As such, if a model is confident about specific program entities (i.e., likely to be faulty), the suspiciousness scores of these program entities would be remarkably higher than those of other entities. Otherwise, all program entities would have uniform suspiciousness scores, indicating that the model lacks confidence in predicting fault entities. Accordingly, we can quantify the suspiciousness scores of all program entities to determine whether some entities have significantly higher scores and select those with the high scores (i.e., high confidence) as pseudo-labels. To achieve this, we employ the Gaussian Mixture Model (GMM) [36] to quantify the suspiciousness scores. Specifically, GMM models variable distributions by decomposing them into several Gaussian distributions. Thus, when a one-component GMM fits the distribution of suspiciousness scores better, it suggests that no particular program entities have significantly higher or lower scores. This indicates that the model lacks strong confidence in any specific entities, making it unsuitable to assign pseudo-labels. In contrast, when a two-component GMM better fits the distribution, it indicates that a subset of program entities has significantly higher suspiciousness scores than the others, allowing us to assign pseudo-labels to those entities with high scores. To determine whether the one-component or two-component GMM better fits the suspiciousness scores, we adopt the Bayesian Information Criterion (BIC) [48], which is widely used to evaluate the performance of GMMs [2, 15, 25]. Based on the BIC, if the two-component GMM is considered more appropriate, we select the program entities from the component with the higher mean suspiciousness scores as pseudo-labels for SSL; Otherwise, we omit pseudo-labeling to avoid introducing noisy pseudo-labels. By utilizing GMM and BIC, we achieve pseudo-labeling without relying on thresholds, while also addressing the challenge of identifying multiple faulty entities in fault localization. Further details on the implementation of our pseudo-labeling mechanism will de discussed in Section 3.3.

3 Design

In this section, we first provide an overview of Legato workflow. Then, we elaborate on the technical details of each stage in Legato.

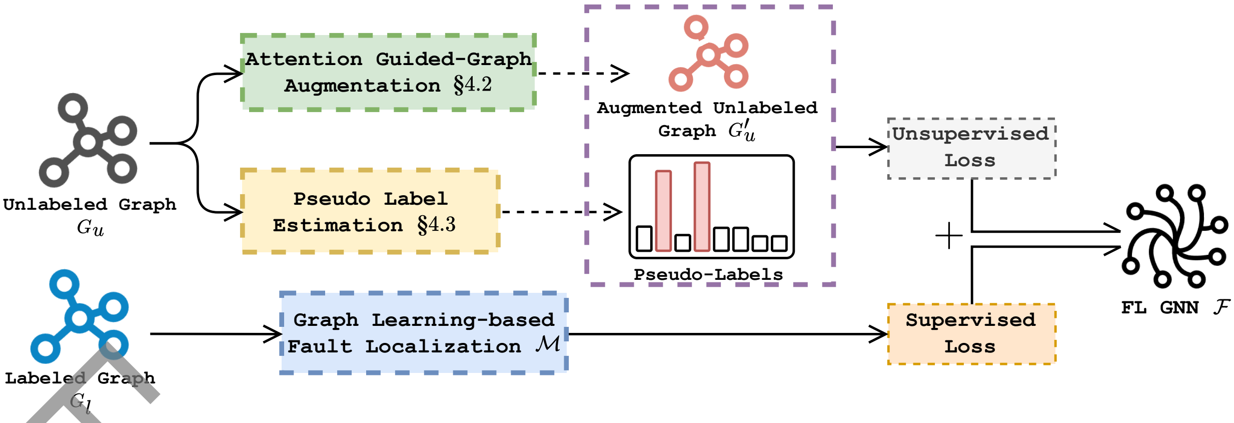
Attention Guided-Graph Augmentation §4.2
Augmented Unlabeled Graph G'_u
Unsupervised Loss
Unlabeled Graph G_u
Pseudo Label Estimation §4.3
Pseudo-Labels
+
FL GNN $\mathcal{F}$
Graph Learning-based Fault Localization $\mathcal{M}$
Supervised Loss
Labeled Graph G_l

Fig. 1. The workflow of LEGATO.

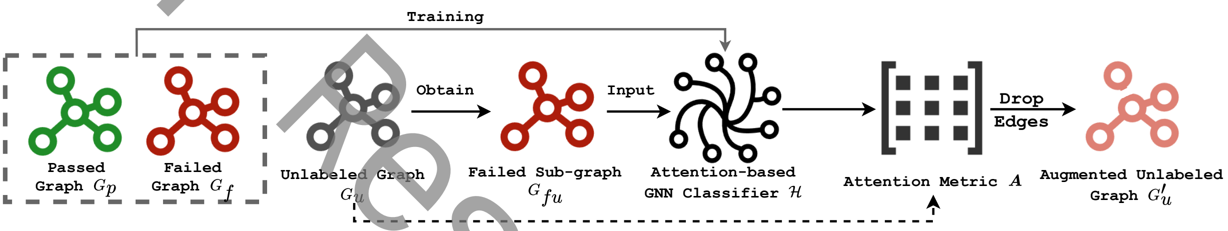

Fig. 2. The workflow of attention-guided graph augmentation.

3.1 Overview

Figure 1 presents the workflow of LEGATO. LEGATO consists of two main components: *attention-guided graph augmentation* and *pseudo label estimation*. The two components work as follows. In the *attention-guided graph augmentation* step, LEGATO utilizes the attention matrix from a pre-trained attention-based GNN designed to classify passed and failed graphs to guide graph augmentation on unlabeled graph G_u. Specifically, we follow our intuition outlined in Section 2.2 to select the edges with the lowest attention weights (i.e., those likely unrelated to faults) and then drop these edges to augment the graph, thereby obtaining the augmented unlabeled graph G'_u. Then, in the *pseudo-label estimation* step, LEGATO quantifies the distribution of suspiciousness scores of G_u to estimate the pseudo-labels. Specifically, for each G_u, LEGATO utilizes the fault localization GNN $\mathcal{F}$ to obtain the suspiciousness score distribution. Then, as discussed in Section 2.2, we leverage one-component and two-component GMMs to respectively fit the suspiciousness score distributions and determine which GMM better quantifies the distribution using the BIC. When the two-component GMM provides a better quantification, we select the program entities belonging to the component with the higher mean suspiciousness scores as pseudo-labels. Otherwise, we do not select pseudo-labels. After pseudo-labeling the unlabeled graph G_u, LEGATO follows the standard SSL paradigm to learn the GBFL methods. More specifically, we train the model by combing the supervised loss $loss_l$ on the labeled data and the unsupervised loss $loss_u$ on the pseudo-labeled data. Particularly, the loss $loss_l$ is defined as $loss_l = \mathcal{L}(\mathcal{F}, G_l, Y_l)$ and the loss $loss_u$ is defined as $loss_u = \mathcal{L}(\mathcal{F}, G'_u, Y_u)$, where $\mathcal{L}$ represent the listwise loss function which is commonly used in the GBFL methods [30, 40], Y_l is the label matrix of G_l, and Y_u is the label matrix generated from pseudo-labels of G_u. Next, we describe each component in detail.

3.2 Attention-Guided Graph Augmentation

In the attention-guided graph augmentation step, Legato inputs the unlabeled graph G_u into the pre-trained attention-based GNN to obtain the attention matrix. Then, guided by the attention matrix, Legato identifies and augments sub-graphs that are more likely fault-unrelated in G_u for graph augmentation. The intuition here is that, after training the attention-based GNN to classify passed and failed graphs, the attention mechanism will reflect the importance of each edge and node during the classification process [32, 50]. In the case of classifying failed graphs, the weights from the attention matrix will increase on sub-graphs that appear exclusively in failed graphs and decrease on sub-graphs that also appear in passed graphs for more effective classification. Thus, the sub-graphs within the failed graphs with lower attention weights are more likely fault-unrelated, and we can perform graph augmentation on them. Furthermore, by training the model to produce similar outputs before and after augmentation, we can make the model focus more on unchanged likely fault-related information and utilize unlabeled data to improve fault localization.

Figure 2 presents the overall pipeline of our proposed attention-guided graph augmentation method. For an unlabeled failed test case present in the unlabeled graph, we first extract its failed sub-graph, consisting of all nodes and edges related to the failed test case. Then, we input the failed sub-graph to a pre-trained attention-based GNN and obtain the corresponding attention matrix. Finally, we utilize the attention matrix to identify the edges with the lowest weights and drop them from the unlabeled graph to produce the augmented unlabeled graph. Note that since we make predictions on the nodes representing program entities, we only augment the edges and not the nodes. In the following, we elaborate on each step in detail.

Attention-based GNN Learning. To identify fault-unrelated sub-graphs for augmentation, we leverage the attention matrix obtained from an attention-based GNN that is designed to classify passed and failed graphs. The insight here is that, for effective graph classification, the attention-based GNN tends to reduce its focus on edges that frequently appear in the passed graphs and are likely fault-free. Transitively, the attention matrix obtained from the GNN provides valuable guidance for identifying sub-graphs unrelated to faults.

The first step in building the attention-based GNN is to construct a training dataset that includes both passed and failed graphs. To achieve this, we iterate through the test cases associated with each graph in the dataset, extracting the relevant nodes and edges to form a new graph. Specifically, if the test case is a passed test case, the extracted graph is labeled as a passed graph; otherwise, it is labeled as a failed graph. Note that we do not need to manually label these graphs as passed or failed because once the test is finished, we can automatically get the result of the test as pass or fail.

After constructing the training dataset, we train the attention-based GNN using the Graph Attention Network (GAT) [51] as our model architecture. The reason why we select GAT is that it has been widely used in various engineering tasks [10, 73] and enables a balance between performance and training time on large-scale dataset [60]. More specifically, let $\mathcal{H}$ denote the GAT. We train $\mathcal{H}$ to classify passed and failed graphs using binary cross-entropy (BCE) to achieve our goal. The loss function is as shown in Eq. 1

$$\mathcal{L}_{BCE} = y_i \log \mathcal{H}(G) + (1 - y_i) \log(1 - \mathcal{H}(G)), \tag{1}$$

where G is a graph, y_i is its label, and $\mathcal{H}(G)$ is the probability that G is failed graph. The labels of passed and failed graphs are 0 and 1, respectively.

Graph Augmentation. With the attention-based GNN, we then use the attention mechanism of the model to rank and augment edges that are more likely to be fault-unrelated. By making the model output consistently before and after such augmentation, we encourage the model to focus on the features that remain invariant, likely related to faults, to improve the effectiveness of fault localization. Specifically, given a failed test case involved in an unlabeled graph, we first obtain

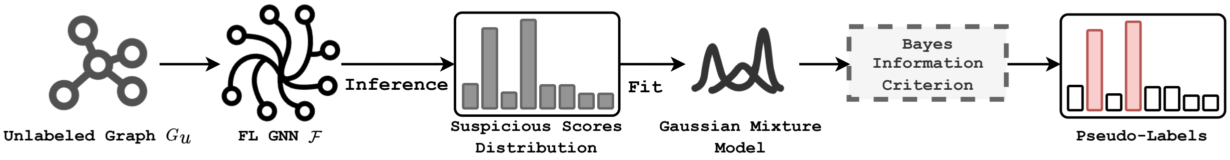

Fig. 3. The workflow of pseudo-label estimation of Legato.

all nodes and edges corresponding to the failed test case to construct a failed sub-graph. We only input the failed sub-graph because it is aligned with the training data format, as the attention-based GNN is trained on passed and failed graphs Then, we input the failed sub-graph into the model to obtain the attention matrix, retaining only the weights associated with the edges. Since we need to make predictions based on program entities (i.e., nodes), we enhance only the edges, not the nodes. Then, We rank all the edges in the graph in ascending order based on their attention weights and select the top p edges to drop.

Formally, let $G_u = \{\mathcal{V}_u, \mathcal{E}_u\}$ represent the unlabeled graph, and let $G_{fu} = \{\mathcal{V}_{fu}, \mathcal{E}_{fu}\}$ represent the failed sub-graph corresponding to the failed test case in G_u, where $\mathcal{V}$ and $\mathcal{E}$ denote the sets of nodes and edges, respectively. The attention matrix extracted from $\mathcal{H}$ is denoted by $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{E}_{fu}|*1}$ and each row in $\boldsymbol{A}$ represent the weight of one edge in $\mathcal{E}_{fu}$. Then, we rank the edges in $\mathcal{E}_{fu}$ based on $\boldsymbol{A}$ to obtain the ascending edge sequence E_r. Next, we select the top p edges from E_r to form the deletion set $\mathcal{E}_d$. Finally, we can obtain the augmented unlabeled graph G'_u by $G'_u = \{\mathcal{V}_u, \mathcal{E}_u \backslash \mathcal{E}_d\}$.

3.3 Pseudo-Label Estimation

As discussed in Section 2.1, the uncertain number of fault entities and their varying suspiciousness scores pose a specific challenge in selecting high-confidence pseudo-labels in fault localization. To address this challenge, in the pseudo-label estimation step, we quantify the distribution of suspiciousness scores of all program entities to assess whether some program entities have significantly higher scores than others. If so, we can select those entities with significantly high suspiciousness scores as pseudo-labels. The key insight of this step is that the suspiciousness scores (i.e., the confidence) of all program entities usually form a probability distribution that sums to one. When some entities have high suspiciousness scores, others will have lower scores. Thus, We can address the challenge by quantifying the distribution of suspiciousness scores across all program entities to select the high-confidence entities without relying on specific thresholds or considering the number of faulty entities.

Figure 3 overviews the pseudo-labeling process. For the unlabeled graph, we first input it into the fault localization GNN to obtain the suspiciousness scores distribution. Then, we quantify the distribution by fitting it into one-component and two-component GMM. Next, we utilize the BIC to evaluate which GMM better quantifies the distribution and whether to select pseudo labels. In the following, we elaborate on each step in detail.

Remark. We conduct pseudo-label estimation on original unlabeled graphs rather than their augmented versions. The reason is that when the training begins, the model has not yet been exposed to augmented versions but has seen the original labeled graphs in supervised learning. Thus, directly estimating pseudo-labels on the model's output on these augmented versions might introduce noisy pseudo-labels, leading to sub-optimal effectiveness.

Suspiciousness Scores Extracting. To quantify suspiciousness scores and select pseudo-labels, we first need to extract the corresponding suspiciousness scores for the unlabeled graphs. This process is aligned with the inference process of the fault localization model. Typically, after a graph is input into a fault localization GNN, the model outputs a logit value for each node. These logits are then normalized using a softmax function to obtain the suspiciousness scores for each node [30, 40].

The suspiciousness scores represent the likelihood of program entities being faulty (i.e., confidence). Specifically, let $\mathcal{F}$ denote the fault localization GNN. Given an unlabeled graph G_u, let $\mathcal{V}_t$ denote the target node set for a failed test case involved in G_u. Next, the logits value of each target node will be normalized by the softmax function as follows:

$$p(v_i) = \frac{\exp\{z_i\}}{\sum_{j=1}^{|\mathcal{V}_t|} \exp\{z_j\}} \tag{2}$$

The $p(v_i)$ is the suspiciousness score of the node v_i. Finally, the suspiciousness scores of all target nodes are denoted as $\mathcal{S} = \{p(v_i)|v_i \in \mathcal{V}_t\}$.

Distribution Quantification. After extracting the suspiciousness scores for all target nodes, we quantify the distribution of these scores to determine whether some program entities have significantly higher suspiciousness scores compared to others, which would indicate the need for pseudo-label selection. The intuition here is that the suspiciousness scores form a probability distribution that sums to one. When some program entities have higher suspiciousness scores (i.e., higher confidence), the suspiciousness scores of the remaining nodes will be relatively lower. Thus, by quantifying the distribution of suspiciousness scores, we can determine whether significantly high suspiciousness program entities exist and select them as pseudo-labels to utilize the unlabeled failed tests, thereby improving fault localization.

Specifically, we use GMM [36] to quantify the distribution and employ the BIC [48] to determine whether to select pseudo labels. GMM is an efficient method for quantifying model output distributions [36]. By specifying different numbers of components, GMM can effectively decompose the distribution and is widely used in various tasks [23, 56]. The BIC is a widely used method for GMM model selection and is better than other criteria for identifying the true model among a set of candidates[1] [2, 15, 25]. The basic idea here is that if the suspiciousness scores of a portion of nodes are significantly higher than those of another portion, using a two-component GMM to decompose the distribution into two distinct distributions will provide a better quantification. Otherwise, a one-component GMM is sufficient to quantify the distribution. We use the BIC to determine whether the one-component or two-component GMM better quantifies the distribution, thereby guiding the decision on whether to select pseudo-labels.

More specifically, we fit the one/two-component GMM on $\mathcal{S}$ to maximize the log-likelihood value by

$$max \sum_{i=1}^{|\mathcal{S}|} \log\left(\sum_{j=1}^{c} \phi_j \mathbb{P}(s_i|u_j, \sigma_j)\right) \tag{3}$$

where s_i is the suspiciousness score of i-th node, c is the number of component, u_j and σ_j are the mean and variance of the j-th component, and ϕ_j denotes the weight of the j-th component and $\sum_{j=1}^{c} \phi_j = 1$. When $c = 1$, fit a one-component GMM, and when $c = 2$, fit a two-component GMM. Once the two GMMs are trained, we utilize the BIC to determine which GMM model better quantifies the distribution by

$$BIC = -2\log(\hat{L}) + \log(|\mathcal{S}|)d \tag{4}$$

where $\hat{L}$ is the maximum likelihood of the model and d is the number of parameters. We only select pseudo-labels and conduct SSL when two-component GMM has the higher BIC score and better fits the distribution. Next, we detailed how we select pseudo-labels.

Pseudo-Label Selection. When the two-component model better fits the distribution, we select the nodes (i.e., program entities) belonging to the component with a higher mean suspiciousness score as our pseudo-labels. Specifically, let u_1 and u_2 be the mean of the two components in the

[1] https://scikit-learn.org/stable/auto_examples/mixture/plot_gmm_selection.html

GMM and satisfy $u_1 > u_2$. Then, we select a node v_i as a pseudo-label (i.e., fault entity) if and only if $\mathbb{P}(s_i|u_1, \sigma_1) > \mathbb{P}(s_i|u_2, \sigma_2)$, where $\mathbb{P}(s_i|u_j, \sigma_j)$ is the probability that the node v_i belong to the j-th component in the GMM. That is to say, v_i is selected as a pseudo-label when it belongs to a component with a larger mean suspiciousness score, indicating that the model has a relatively high confidence in it. As such, we can obtain a pseudo-labels set $\boldsymbol{P}_u = \{v_i|\mathbb{P}(s_i|u_1, \sigma_1) > \mathbb{P}(s_i|u_2, \sigma_2) \wedge v_i \in \mathcal{V}_t\}$ for unlabeled graph G_u. After obtaining pseudo-labels for a given unlabeled failed test case in G_u, we combine them with the augmented unlabeled failed graph for same failed test case and employ listwise loss function to train the model.

3.4 Training with Legato

Algorithm 1: Semi-Supervised Learning Algorithm with Legato

```
Input: Labeled Graph set $\mathcal{G}_l$, Unlabeled Graph set $\mathcal{G}_u$, Learning Epoch $E$, GNN-based Fault Localization Technique $\mathcal{M}$, drop rate $p$,
       Scalar hyperparameter $\lambda_u$
Output: Fault Localization Model $\mathcal{F}$
1  $\mathcal{F} \leftarrow$ InitModel ();
2  $e \leftarrow 0$;
3  $\mathcal{H} \leftarrow$ AttentionLearning ($\mathcal{G}_l, \mathcal{G}_u$) // Train the attention-based GNN on all extracted passed and failed graphs
4  while $e < E$ do
5      $loss_l \leftarrow \mathcal{M}(\mathcal{F}, \mathcal{G}_l)$ ;
6      $loss_u \leftarrow 0$ ;
7      for each $G_u \in \mathcal{G}_u$ do
8          for each failed test case $c$ involved in $G_u$ do
               /* Attention-guided graph augmentation */
9              Extract the failed sub-graph $G_{fu}$ corresponding to $c$;
10             Obtain the attention weight matrix $\boldsymbol{A}$ from $\mathcal{H}$ when classify $G_{fu}$ ;
11             Rank the edges in $G_{fu}$ based on $\boldsymbol{A}$ to get the ascending edge sequence $E_r$ ;
12             Select top $p$ edges in $E_r$ and drop them from $G_u$ to obtain $G'_u$;
               /* Pseudo-label estimation */
13             Obtain suspiciousness scores $\mathcal{S}$ via Eq. 2 for all target nodes in $c$;
14             Fit $\mathcal{S}$ into two GMMs via Eq. 3 and obtain the best model $G$ by BIC via Eq. 4 ;
15             if $G$ is two-component GMM then
16                 Select the nodes from the component with higher mean suspiciousness scores as pseudo-labels $\boldsymbol{P}_u$;
17                 Construct label matrix $Y_u$ from pseudo-labels $\boldsymbol{P}_u$ ;
18                 Add the listwise loss value on $G'_u$ and $Y_u$ to $loss_u$ ;
19             end
20         end
21     end
22     $loss = loss_l + \lambda_u loss_u$;
23     Update parameter of model $\mathcal{F}$ by backpropagation on $loss$;
24 end
25 return $\mathcal{F}$
```

After describing the two steps of Legato, we further demonstrate how Legato integrates with the GBFL for SSL through Algorithm 1. In this algorithm, we first train the attention-based GNN to classify passed and failed graphs for attention-guided graph augmentation (Line 3). Then, during the training process (Lines 4-24), we first obtain the supervised loss $loss_l$ from the GBFL technique $\mathcal{M}$ (Line 5). After that, Legato iterates through each unlabeled graph $G_u \in \mathcal{G}_u$ and each failed test case c in G_u (Lines 7-21) to perform attention-guided graph augmentation (Lines 9-12) and pseudo-label estimation (Lines 13-19). Specifically, we first obtain the failed sub-graph G_{fu} corresponding to c and input it into $\mathcal{H}$ to obtain the attention matrix $\boldsymbol{A}$ (Lines 9-10). Next, we rank the edges from G_{fu} guided by weights from $\boldsymbol{A}$ in ascending order and obtain the ascending edge sequence E_r (Line 11). finally, we get the augmented unlabeled graph G'_u for failed test case c by selecting the top p edges in E_r and dropping them from G_u (Line 12). After augmentation, we conduct pseudo-label estimation (Lines 13-19). We first obtain the suspiciousness scores $\mathcal{S}$ of all target nodes of c via

Table 1. Statistics about the dataset.

Attribute	Value	Attribute	Value	Attribute	Value
Failed #Tests	7500	Labeled Failed #Tests	600	Unlabeled Failed #Tests	6900
Passed #Tests	7500	Average #Files	367.27	Average #Methods	1085.28

Eq. 2 (Line 13). Then, we fit the scores into one/two-component GMMs and obtain the best model by BIC via Eq. 3 and Eq. 4 (Line 14). If the best model is a two-component GMM, we further select the nodes from the higher mean suspiciousness scores as pseudo-labels $\boldsymbol{P}_u$ (Lines 15-16). Otherwise, we do not select the pseudo-labels and do not calculate unsupervised loss $loss_u$ on it. Then, let $\boldsymbol{Y}_u = \{0, 1\}^{|\mathcal{V}_u|\times 1}$ denote the label matrix of G_u, the label $y_i \in \boldsymbol{Y}_u$ of node v_i is set to 1 if $v_i \in \boldsymbol{P}_u$ and otherwise, set to 0 (Line 17). Next, we calculate the loss value on G'_u and $\boldsymbol{Y}_u$ from the listwise loss function denoted by $\mathcal{L}_{list} = -\sum_{i=1}^{n} y_i \log(p(v_i))$, where $p(v_i)$ denotes the suspiciousness score of node v_i and n is the number of target nodes (Line 18). Finally, the loss minimized by Legato is $loss_l + \lambda_u loss_u$, where λ_u is a fixed scalar hyper-parameter denoting the relative weight of the unlabeled loss [47] (Lines 22-23).

4 Evaluation

We evaluate Legato on the following research questions:

- RQ1: How does Legato's effectiveness compare to that of state-of-the-art SSL techniques on fault localization?
- RQ2: How does Legato perform in the cross-project prediction scenario?
- RQ3: How do different components within Legato affect its overall effectiveness?
- RQ4: What is the quantity and quality of pseudo-labels we generated?
- RQ5: How does the training efficiency of Legato compare to that of SSL techniques?

4.1 Evaluation Setup

Benchmark. To answer the RQs, we collaborated with our industrial partner which operates multiple digital product lines worldwide. They provided us with tests generated during system testing of eleven software that cover diverse product lines and platforms. Each software written by C/C++ exceeds one million lines of code, averaging hundreds of packages, thousands of files, and tens of thousands of methods. Table 1 reports key statistics about the dataset. We have a total of 15,000 tests, with 7,500 failed tests and 7,500 passed tests. Following the commonly used SSL setup [6, 47, 55], engineers from our industrial partner randomly label 8% of the failed tests, with each labeled test involving multiple fault entities. Note that when calculating the percentage of labeled data, we only consider failed tests, as passed tests inherently do not require labeling. We did not evaluate Legato on defects4j (v2.0.0) because the dataset contains only a small number of failed tests (1,486). Under the semi-supervised setup with 8% labeled samples, we had only 118 labeled samples, with an average of 8.48 labeled samples per project. There are too few tests available for learning a meaningful model [24, 47]. Meanwhile, SSL is highly compatible with industrial settings, as software typically undergoes extensive testing before release, generating many unlabeled failed tests. Thus, given limited resources, engineers can only label a small portion of failed tests, making SSL a suitable approach. When the number of tests is small, we can directly label them all and perform supervised learning rather then SSL.

Baselines. To the best of our knowledge, we are the first to attempt SSL for fault localization, whereas SSL is typically evaluated on image classification [4, 21, 47]. Consequently, we selected three representative state-of-the-art SSL techniques that could be adapted to GBFL as baselines, which are:

- Π-Model [41] uses data augmentation and dropout mechanisms to perturb unlabeled samples and trains the model by ensuring consistent model outputs for these perturbations.
- Pseudo-Labeling [21] selects the highest confidence predictions from the model's outputs on unlabeled samples as pseudo-labels and uses these for further training the model.
- FixMatch [47] initially obtains pseudo-labels using weak augmented data and a fixed threshold and then trains the model on strong augmented data and these pseudo-labels to conduct SSL.

Base GBFL techniques. SSL extends an existing supervised learning algorithm to utilize both labeled and unlabeled data. In our evaluation, we consider the highly representative and state-of-the-art techniques GRACE [30] and DepGraph [40] as the base supervised learning algorithm. Unless specifically stated, the experiments were conducted using GRACE as a base GBFL. Specifically, we follow the same approach as theirs to construct the graph. We can only connect the test node with the root node of the AST of the covered method since obtaining line coverage information through instrumentation has a significant impact on testing efficiency in the context of testing of large-scale software, which typically involves thousands of methods.

Evaluation Metric. Following previous work [24, 30, 40], we adopt Recall at Top-N (N=1,3,5), MFR (Mean First Rank), and MAR (Mean Average Rank) as our evaluation metrics. *Recall at Top-N* measures the proportion of all failed tests in which at least one faulty entity is ranked within the top N positions. *MFR* measures the mean rank of the first faulty element across all failed tests. *MAR* is the mean of the average ranking of all the faulty entities of all failed tests. Higher Recall at Top-N and lower MFR and MAR suggest that the faulty entities are located closer to the top of the ranked list, indicating better localization performance. We use the *worst* ranking for the tied elements that have the same suspiciousness scores.

Localization Level. We only perform fault localization at the file and method levels because we cannot obtain the line coverage information as discussed above. For a method, the suspiciousness score is obtained by applying the softmax function to normalize the logit value output by the model for that method (cf. Eq. 2), consistent with [30, 40]. For a file, we select the highest suspiciousness score among all methods within the file to represent the file's suspiciousness score, consistent with [46].

Within-project and Cross-Project Setting. We explore Legato's performance in within-project and cross-project settings in RQ1 and RQ2, respectively. In the within-project scenario, the data coming from the same project populates both training and testing sets. Due to the large volume of data and the cost of model training, we employed ten-fold cross-validation in within-project settings instead of the leave-one-out validation method used in previous work [24, 30]. In practice, a project might not have enough historical data to conduct model training. Hence, we further investigate the effectiveness of different techniques in cross-project settings. Consistent with existing literature [28], we perform fault localization on each failed test in a project while training a model on the tests from all the remaining projects and repeat the process for each project.

Implementation. We build Legato based on PyTorch Geometric [12] and Scikit-learn [35]. We utilize the tree-sitter [1] to obtain the AST of methods. For the parameters in Legato, We employed grid search to determine the optimal parameter combination. Furthermore, all experiments are conducted with a fixed random seed to reduce variations across the experiments.

For the baselines, we directly adopt their open-source implementations and empirically tune the hyper-parameters by grid search because the original hyper-parameters haven't been tested on our task. Note that both Π-Model and FixMatch involve data augmentation, originally applied to image data in their papers, which differs from the graph data we use. Therefore, we adopted a widely used edge centrality-based graph augmentation [75] for them, which identifies important edges via

Table 2. Comparison with state-of-the-art at file and method levels using GRACE as base GBFL.

Techniques	Method Level					File Level				
	Top-1 (↑)	**Top-3 (↑)**	**Top-5 (↑)**	**MFR (↓)**	**MAR (↓)**	**Top-1 (↑)**	**Top-3 (↑)**	**Top-5 (↑)**	**MFR (↓)**	**MAR (↓)**
GRACE	40.46	49.32	54.02	52.08	77.78	52.88	61.13	66.09	21.52	33.72
Π-Model	44.61	53.88	61.05	49.46	66.85	56.84	67.42	76.45	12.79	22.97
Pseudo-Labeling	51.02	60.97	66.27	32.91	52.43	58.06	70.03	85.48	8.46	11.74
FixMatch	58.37	66.28	70.35	22.05	35.56	64.35	73.42	80.43	6.11	9.90
Legato	**68.91**	**77.69**	**79.66**	**11.74**	**17.49**	**79.03**	**83.71**	**90.32**	**2.37**	**4.32**

Table 3. Comparison with state-of-the-art at file and method levels using DepGraph as base GBFL.

Techniques	Method Level					File Level				
	Top-1 (↑)	**Top-3 (↑)**	**Top-5 (↑)**	**MFR (↓)**	**MAR (↓)**	**Top-1 (↑)**	**Top-3 (↑)**	**Top-5 (↑)**	**MFR (↓)**	**MAR (↓)**
DepGraph	43.32	50.16	59.33	50.12	68.33	57.63	64.88	71.24	18.34	25.67
Π-Model	47.82	58.28	65.19	44.25	60.12	59.44	69.17	78.32	10.91	18.33
Pseudo-Labeling	55.89	62.87	69.45	26.29	40.60	61.82	72.88	86.81	8.11	11.42
FixMatch	61.52	67.35	72.61	20.18	32.47	68.39	74.02	82.92	5.74	9.42
Legato	**70.33**	**78.44**	**83.53**	**10.22**	**16.87**	**82.26**	**85.44**	**90.77**	**2.16**	**4.09**

node centrality measures (i.e., the node degree). Then, it adaptively drops edges by giving large removal probabilities to unimportant edges to highlight important connective structures.

Due to the spatial constraints, all hyper-parameters and detailed configuration of Legato and baselines have been included on our project website. All experiments are conducted on a workstation with AMD Ryzen 9 3900XT, 32GB memory, and two RTX 4090 GPUs, running Ubuntu 20.04.

4.2 RQ1: Effectiveness of Legato

The primary objective of this RQ is to evaluate the effectiveness of Legato in semi-supervised fault localization with only a small subset of failed tests labeled (8%). We conduct a comparative analysis of Legato against three SSL baselines outlined in Section 4.1 on two different representative base GBFL, examining performance at both method and file levels. Results are detailed in Table 2 and Table 3. Furthermore, we also compared the effectiveness of different methods at the file level when there were fewer failed labeled tests. In this experiment, we randomly take a portion of the labeled samples to keep and discard the labels in the remaining labeled samples, treating them as unlabeled. The comparison results are shown in Figure 4. Figure 4 demonstrates the effectiveness of different methods when only 1% to 8% of failed samples are labeled. The x-axis represents the percentage of available labeled samples, and the y-axis shows various metrics.

Overall Effectiveness. From the Table 2 and Table 3, we have the following observations. Firstly, Legato can effectively leverage unsupervised samples to enhance the performance of existing GBFL methods when only a small portion of labeled samples is available. Notably, when applied to GRACE and DepGraph as the base GBFL models, Legato achieved Top-1 accuracy of 68.91% and 70.33% at the method level, representing significant improvements of 70.32% and 62.35%, respectively, over using GRACE and DepGraph alone. Furthermore, the MFR and MAR metrics exhibit substantial improvement, with Legato yielding an average improvement of 78.53% in MFR and 76.41% in MAR at the method level compared to GRACE and DepGraph, respectively. Similar trends are observed at the file level. Specifically, at the file level, Legato achieved average improvements of 46.10%, 34.31%, 32.03%, 88.61%, and 85.63% over the two base GBFL models in terms of Top-1,

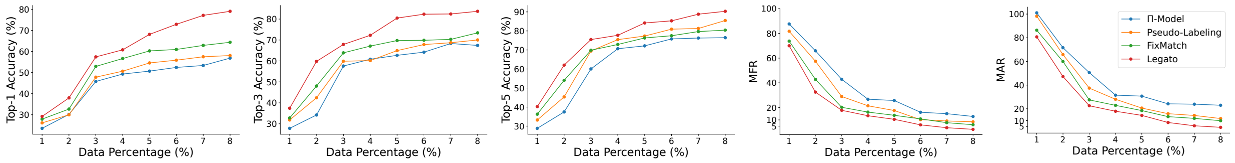

Fig. 4. Comparison with state-of-the-art at file level under different labeling rates.

Top-3, Top-5 accuracy, MFR, and MAR, respectively. These results demonstrate that LEGATO can effectively enhance the performance of existing GBFL methods and remains stable across various GBFL approaches.

Second, from Table 2 and Table 3, we can observe that LEGATO consistently outperforms studied SSL baselines across all five metrics at both method and file levels on different GBFL methods. Specifically, the method level improvements of LEGATO over all the SSL baselines on GRACE and DepGraph are remarkable, ranging from 14.36% to 54.47% on Top-1, ranging from 16.47% to 44.19% on Top-3, ranging from 13.23% to 30.48% on Top-5, from 46.76% to 76.90% on MFR, and from 48.04% to 73.83% on MAR. Similar trends are observed at the file level, underscoring LEGATO's effectiveness compared to other studied SSL baselines across different localization level and GBFL methods.

Third, the comparison between LEGATO and FixMatch—both of which leverage pseudo-labels and augmentation techniques—reveals that LEGATO markedly outperforms FixMatch at both localization levels and on different base GBFL. Although the edge centrality-based graph augmentation used by FixMatch is widely applied in various graph learning scenarios, the centrality measures do not effectively capture fault information in the test graphs. In contrast, LEGATO trains an attention network using passed and failed test graphs, allowing the attention mechanism to identify edges in the failed graph that are more likely to be irrelevant to faults. This makes the model focus more on fault-related features unchanged during augmentation, which enhances fault localization. The improvement also suggests that our pseudo-label estimation is more effective than the fixed threshold employed by FixMatch. The estimation step in LEGATO allows us to select all program entities with significantly high suspiciousness scores as pseudo-labels, avoiding the threshold selection and addressing the issue of multiple fault entities.

Different percentage of labeled data. From the figure 4, we can observe that as the number of available labeled samples increases, the performance of all methods improves across different metrics. Specifically, LEGATO achieves the best effectiveness across all percentages. In scenarios with very few labeled data, the effectiveness of models trained by all methods significantly declines. It is reasonable that when the amount of labeled data is extremely limited, the supervised learning component within semi-supervised learning may be unable to train the model on labeled data, making fault localization challenging. When the model performs poorly, its outputs for unlabeled samples may be incorrect, significantly impacting the effectiveness of all SSL methods. However, LEGATO still manages to achieve 59.8% Top-3 accuracy with only 2% of labeled samples, which is a 24.5% improvement over the best performance baseline. This indicates that LEGATO is better adapted and more robust to an environment with minimal labeled data, further underscoring its effectiveness.

To further confirm the observations above, we have followed previous works [24, 30] to perform the Wilcoxon signed-rank test [57] with Bonferroni corrections [11] to investigate the statistical significance between LEGATO and other baselines. The results show that LEGATO is significantly better than all studied baselines at the significance level of 0.05.

Table 4. Cross-project effectiveness at file and method levels.

Techniques	**Method Level**					**File Level**				
	Top-1 (↑)	**Top-3 (↑)**	**Top-5 (↑)**	**MFR (↓)**	**MAR (↓)**	**Top-1 (↑)**	**Top-3 (↑)**	**Top-5 (↑)**	**MFR (↓)**	**MAR (↓)**
GRACE	25.06	30.41	34.77	114.57	133.25	33.82	38.14	42.78	47.24	55.98
Π-Model	27.78	33.65	38.74	92.63	110.37	37.95	45.39	53.11	26.34	35.44
Pseudo-Labeling	26.32	31.22	36.53	107.2	123.64	37.12	42.59	49.82	27.11	39.88
FixMatch	31.58	37.72	44.05	87.74	103.29	40.17	52.97	57.45	22.66	25.67
Legato	**43.84**	**46.39**	**51.63**	**64.95**	**79.32**	**52.68**	**57.64**	**60.27**	**14.42**	**19.18**

Table 5. The effectiveness of different variants at method level.

Variants	**Top-1 (↑)**	**Top-3 (↑)**	**Top-5 (↑)**	**MFR (↓)**	**MAR (↓)**
Legato$_{PLE}$	49.27	56.94	64.82	46.33	60.91
Legato$_{EA}$	57.31	64.41	73.58	30.24	42.55
Legato$_{AA}$	52.28	55.83	61.44	44.7	67.91
Legato$_{PL}$	55.97	62.33	70.44	24.62	37.17
Legato$_{GMM}$	60.49	70.58	71.27	19.85	31.83
Legato	**68.91**	**77.69**	**79.66**	**11.74**	**17.49**

4.3 RQ2: Cross-project Effectiveness of Legato

In RQ2, we further evaluate the method and file level effectiveness of Legato in a cross-project scenario. Table 4 presents the comparison results between Legato and other baselines in the cross-project scenario. From the tables in within-project and cross-project scenarios, we observed a certain degree of decline in the effectiveness of all methods. This decline is reasonable because of the significant differences in fault contexts between the tested project and other projects in cross-project scenarios, which can only be learned from other projects and unlabeled data. Thus, the cross-project scenario presents significant challenges to how SSL methods utilize unlabeled data, as they employ the same GNN-based FL method. Despite the performance degradation of Legato, it still performs better than other baselines. Specifically, the improvements of Legato compared with the second-best baselines achieve 38.82%, 22.98%, 17.29%, 25.97%, and 23.20% in terms of Top-1, Top-3, Top-5, MFR and MAR, respectively. Similar trends are observed at the file level. Additionally, the performance drop of Legato in the cross-project scenario is also smaller than the baselines. For example, in terms of Top-1 accuracy, Legato decreased by 36.28% and 33.34% at the method and file level, which is less than the 41.61% and 37.57% decrease experienced by FixMatch. These results demonstrate the effectiveness of Legato in the cross-project scenario and suggest that Legato's attention-based graph augmentation enables the model to better utilize passed and failed graphs to identify potential fault entities in cross-project settings.

4.4 RQ3: Ablation Study

In this RQ, we conduct a series of ablation studies on different graph augmentation and pseudo-label estimation policies to further analyze the impact of each step in Legato. Specifically, we consider the following five variants of Legato: Legato$_{PLE}$ only employs pseud-label estimation. Legato$_{EA}$ replaces the attention-guided graph augmentation to edge centrality-based graph augmentation [75], which has been adapted in baselines and is widely used in graph learning. Legato$_{AA}$ only employs attention-guided graph augmentation. Legato$_{PL}$ replaces the pseudo-labels estimation policies with pseudo-labeling [21], which selects the entity with the highest confidence as pseudo-label. Legato$_{GMM}$ removes the BIC and only employs the two-component GMM in pseudo-labels estimation. Table 5 summarizes the study results, and we have the following observations.

Table 6. The coverage and impurity at the end of the training of different methods involved pseudo-labels.

Techniques	Coverage (↑)	Impurity (↓)
Pseudo-Labeling	56.47%	29.92%
FixMatch	80.25%	37.56%
Legato	77.14%	21.06%

First, removing any component from Legato results in a noticeable decline in effectiveness. Legato$_{PLE}$ and Legato$_{AA}$ are the least effective variants among all. Specifically, the Top-1 accuracy of Legato$_{PLE}$ and Legato$_{AA}$ decrease by 28.50% and 24.13% compared to Legato respectively This indicates that both our attention-guided graph augmentation and pseudo-label estimation significantly contribute to the effectiveness of Legato. Additionally, this suggests that combining augmentation and pseudo-labels can achieve greater effectiveness in fault localization than employing only one of them.

Second, during pseudo-label estimation, we found that choosing entities with the highest suspiciousness scores as pseudo-labels (Legato$_{PL}$) or using only a 2-component GMM (Legato$_{GMM}$) resulted in sub-optimal effectiveness. In particular, the Top-5 accuracy of Legato$_{PL}$ and Legato$_{GMM}$ decrease by 11.57% and 10.53% compared to Legato respectively. This indicates that directly selecting program entities with high suspiciousness scores as pseudo-labels without quantifying the entire distribution may more easily introduce noisy labels, leading to sub-optimal effectiveness. By combining GMM with BIC, we can dynamically determine during training if some program entities' suspiciousness scores are significantly higher than others, thus estimating pseudo-labels without fixed thresholds and minimizing the introduction of noisy pseudo-labels.

Finally, in terms of graph augmentation, we observed a decrease in model effectiveness when replacing attention-guided graph augmentation with edge centrality-based graph augmentation. Specifically, the Top-1 accuracy of Legato$_{EA}$ was 16.83% lower than that of Legato. This result further demonstrates the effectiveness of our attention-guided graph augmentation. This is because edge centrality-based graph augmentation methods assess the importance of edges based solely on the centrality of nodes. However, the node centrality does not necessarily reflect its relevance to faults, so the augmentation process may affect the fault context, leading to sub-optimal effectiveness. In contrast, our method uses an attention mechanism to target augmentations on subgraphs that are more likely to be irrelevant to faults, thereby enhancing the performance of Legato.

4.5 RQ4: The Quantity and Quality of Pseudo-labels

To better understand the role of pseudo-labels in Legato and baselines, we conducted both quantitative and qualitative analyses of the generated method-level pseudo-labels. Specifically, we randomly selected 200 unlabeled failed tests from the dataset and had engineers from our industrial partner label these tests. Following previous work [47], we collected the pseudo-labels generated by different methods on these 200 unlabeled failed tests after training and compared them to the ground truth labels. We define two additional metrics: the **Coverage** (the rate of true labels that are covered by the pseudo-labels) and **Impurity** (the rate of incorrect labels within the pseudo-labels), which are computed as follows:

$$\text{Coverage} = \frac{\sum_{i=1}^{N} \frac{|P_i \cap L_i|}{|L_i|}}{N}, \quad \text{Impurity} = \frac{\sum_{i=1}^{N} \frac{|P_i \setminus L_i|}{|P_i|}}{N} \tag{5}$$

where P_i and L_i represent the pseudo-labels and true labels for the i-th test, respectively, and N is the number of unlabeled failed tests. Table 6 presents the coverage and impurity of all methods involving pseudo-labels. From the table, we can observe that Legato achieves a better balance

Table 7. Training time cost of studied SSL methods.

Techniques	Π-Model	Pseudo-Labeling	FixMatch	Legato
Training Time	2h51mins	2h10mins	3h7mins	4h41mins

between coverage and impurity. Notably, while achieving the lowest impurity, Legato's coverage is only 3.87% lower than that of FixMatch. This is because our pseudo-label estimation does not rely on a specific threshold but instead makes the estimation based on the overall distribution of suspiciousness scores across program entities. Thus, we can reduce the introduction of noisy labels while covering more true faulty entities. Although FixMatch achieves slightly higher coverage than Legato, it has the worst impurity, as its reliance on a fixed threshold struggles to handle uncertainty in the number of faulty entities and variations in their confidence. Pseudo-Labeling, on the other hand, has lower coverage because it always selects the label with the highest confidence as the pseudo-label, which tends to introduce noisy labels when the confidence across all entities is low, ultimately reducing the effectiveness of the model.

4.6 RQ5: Efficiency of Legato

This RQ empirically analyzes the efficiency of Legato. Table 7 presents the time cost of training a model using Legato and the studied SSL approaches. As shown in this table, compared to the studied baselines, Legato requires more time to train the model because Legato requires pre-trained the attention-based GNN in graph augmentation and fit one/two-component GMM and BIC to perform pseudo-label estimation. Nevertheless, it is important to emphasize that Legato achieves significantly better performance in locating faults than the baselines, as demonstrated in Section 4.2. Therefore, we believe such a time cost of Legato is worthwhile for achieving better models. Furthermore, considering that the training process is offline, the training time of Legato is also acceptable in practice.

5 Discussion

Threats to validity. The main threat to *internal validity* lies in the technical implementation of Legato, the compared approaches, and experimental scripts. To mitigate this threat, we developed Legato based on widely used libraries and employed the original implementations of the compared approaches. We have reviewed the source code of Legato and experimental scripts thoroughly. The main threat to *external validity* may be tied to the benchmark used in our study. To reduce this threat, we perform experiments on an industrial dataset provided by our industrial partner, which contains 7,500 passed and 7,500 failed tests (8% of failed tests are labeled) from 11 large-scale software, each exceeding one million lines of code, and covering diverse product lines and platforms. Furthermore, we compare Legato with 3 representative and state-of-the-art semi-supervised learning baselines in our experiments. The main threat to *construct validity* lies in the parameters in Legato and metrics used in experiments. To mitigate this threat, we present the detailed parameter settings on our project site. To reduce the threat from metrics employed, we employed various metrics that are widely used in prior fault localization studies [24, 30].

Generalizability of Legato. Legato has demonstrated remarkable effectiveness in learning fault localization GNN when only a small portion of graphs are labeled. Nevertheless, we believe its key insights can also be further generalized to other LBFL methods in the future. The generalizability of Legato comes from two aspects. First, the key insight of augmentation in Legato is that the attention mechanism can be trained to capture the likely fault-unrelated information. This idea does not involve specific types of inputs and can be utilized in other LBFL methods by training the attention mechanism on the features from passed and failed tests. Second, the key insight of our pseudo-label estimation is to select pseudo-labels by quantifying the relative magnitudes of suspiciousness scores among different program entities. This method only relies on the suspiciousness scores, which can be extracted from all LBFL methods and models. For future work, we plan to extend Legato to various LBFL methods.

6 Related work

Learning-based fault localization. Deep learning technologies have found extensive application in fault localization [20, 24, 26, 28, 30, 31, 40, 46, 65, 69, 72, 76]. Learning-based fault localization approaches can be divided into two categories: one category improves feature representation from the extracted test data through learning methods. For instance, GRACE [30] represents coverage using a graph structure and employs GGNN [27] for learning and calculating suspiciousness scores. The other category combines features across different dimensions through learning methods. For instance, FLUCCS [46] utilizes suspiciousness values derived from 33 SBFL formulae, combined with code age, churn, and complexity, as diverse features for the Support Vector Machine to process. Among the LBFL techniques, the graph learning-based fault localization (GBFL) has been extensively researched and achieved promising results recently [22, 30, 38–40, 59, 64]. These techniques model source code or other features using graphs, enabling the model to better learn how to represent or combine these features for improved fault localization. In this paper, we build Legato based on the GBFL techniques, aiming to enhance their effectiveness within insufficient labeled data.

Semi-supervised Learning. Semi-supervised learning (SSL) is a machine learning approach that utilizes both labeled and unlabeled data to improve learning accuracy [49]. Semi-supervised learning methods can broadly be divided into three categories: wrapper methods, unsupervised preprocessing, and intrinsically semi-supervised approaches. Wrapper methods extend supervised learning to semi-supervised by utilizing labeled data to supervise the training of models and using the model's output on unlabeled samples to further train the model. Representative techniques include pseudo-labeling [3, 17, 21, 34, 66] and co-training [53, 74]. Unsupervised preprocessing involves processing unlabeled samples in an unsupervised manner and using the processed features to enhance the training of supervised models. Notable techniques include feature extraction [42, 44] and cluster-then-label [5, 14]. Intrinsically semi-supervised methods directly integrate unlabeled samples into the loss function of supervised learning to extend it to semi-supervised learning. The most representative perturbation-based methods assume that the predictions for the augmented and the original versions of the same inputs should be similar [4, 19, 41, 43]. In recent years, combining consistency regularization and pseudo-labeling has led to significant successes in semi-supervised learning [8, 18, 47, 55]. These methods use the model's outputs on unlabeled samples as pseudo-labels, then train the model on these pseudo-labeled and augmented unlabeled samples to achieve semi-supervised learning. Our approach falls into this category.

7 Conclusion

In this paper, we propose Legato, a novel semi-supervised learning framework for GNN-based fault localization, designed to effectively locate faults within the context of insufficient labeled failed tests and a large number of unlabeled failed tests. Legato adopts a two-step approach for utilizing unlabeled failed tests. In the attention-guided graph augmentation, Legato leverages the attention mechanism to identify and augment the more likely fault-unrelated edges and obtain the augmented unlabeled graphs to conduct consistency regularization. Then, in the pseudo-label estimation step, Legato employs GMM and BIC to determine the pseudo-labels of unlabeled graphs to eliminate the optimal threshold selection and address the challenge arising from multi-fault entities in pseudo-label estimation. The experimental results demonstrate the effectiveness of Legato in fault localization within insufficient labeled failed tests.

8 Data Availability

Our code and configuration are publicly available at https://github.com/pppppkun/Legato.

Acknowledgments

This research is supported by the National Natural Science Foundation of China under Grant Nos. 62372227 and 62402214 and the Natural Science Foundation of Jiangsu Province under Grant No. BK20241194.

References

[1] 2024. Tree-sitter. https://tree-sitter.github.io/tree-sitter/

[2] Donald WK Andrews. 1999. Consistent moment selection procedures for generalized method of moments estimation. *Econometrica* 67, 3 (1999), 543–563. https://doi.org/10.1111/1468-0262.00036

[3] Eric Arazo, Diego Ortego, Paul Albert, Noel E. O'Connor, and Kevin McGuinness. 2020. Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning. In *2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020*. IEEE, 1–8. https://doi.org/10.1109/IJCNN48605.2020.9207304

[4] Philip Bachman, Ouais Alsharif, and Doina Precup. 2014. Learning with Pseudo-Ensembles. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger (Eds.). 3365–3373. https://proceedings.neurips.cc/paper/2014/hash/66be31e4c40d676991f2405aaecc6934-Abstract.html

[5] Kristin P. Bennett and Ayhan Demiriz. 1998. Semi-Supervised Support Vector Machines. In *Advances in Neural Information Processing Systems 11, [NIPS Conference, Denver, Colorado, USA, November 30 - December 5, 1998]*, Michael J. Kearns, Sara A. Solla, and David A. Cohn (Eds.). The MIT Press, 368–374. http://papers.nips.cc/paper/1582-semi-supervised-support-vector-machines

[6] David Berthelot, Nicholas Carlini, Ian J. Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. 2019. MixMatch: A Holistic Approach to Semi-Supervised Learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 5050–5060. https://proceedings.neurips.cc/paper/2019/hash/1cd138d0499a68f4bb72bee04bbec2d7-Abstract.html

[7] An Ran Chen, Tse-Hsun (Peter) Chen, and Junjie Chen. 2022. How Useful is Code Change Information for Fault Localization in Continuous Integration?. In *37th IEEE/ACM International Conference on Automated Software Engineering, ASE 2022, Rochester, MI, USA, October 10-14, 2022*. ACM, 52:1–52:12. https://doi.org/10.1145/3551349.3556931

[8] Hao Chen, Ran Tao, Yue Fan, Yidong Wang, Jindong Wang, Bernt Schiele, Xing Xie, Bhiksha Raj, and Marios Savvides. 2023. SoftMatch: Addressing the Quantity-Quality Trade-off in Semi-supervised Learning. *CoRR* abs/2301.10921 (2023). https://doi.org/10.48550/ARXIV.2301.10921 arXiv:2301.10921

[9] Wei Chen, Wu Chen, Jiamou Liu, Kaiqi Zhao, and Mingyue Zhang. 2023. SupConFL: Fault Localization with Supervised Contrastive Learning. In *Proceedings of the 14th Asia-Pacific Symposium on Internetware, Internetware 2023, Hangzhou, China, August 4-6, 2023*, Hong Mei, Jian Lv, Zhi Jin, Xuandong Li, Xiaohu Yang, and Xin Xia (Eds.). ACM, 44–54. https://doi.org/10.1145/3609437.3609441

[10] Xiao Cheng, Haoyu Wang, Jiayi Hua, Guoai Xu, and Yulei Sui. 2021. DeepWukong: Statically Detecting Software Vulnerabilities Using Deep Graph Neural Network. *ACM Trans. Softw. Eng. Methodol.* 30, 3 (2021), 38:1–38:33. https://doi.org/10.1145/3436877

[11] Olive Jean Dunn. 1961. Multiple comparisons among means. *Journal of the American statistical association* 56, 293 (1961), 52–64. https://doi.org/10.2307/2282330

[12] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[13] Ariel Gera, Alon Halfon, Eyal Shnarch, Yotam Perlitz, Liat Ein-Dor, and Noam Slonim. 2022. Zero-Shot Text Classification with Self-Training. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, 1107–1119. https://doi.org/10.18653/V1/2022.EMNLP-MAIN.73

[14] Andrew B. Goldberg, Xiaojin Zhu, Aarti Singh, Zhiting Xu, and Robert D. Nowak. 2009. Multi-Manifold Semi-Supervised Learning. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009 (JMLR Proceedings, Vol. 5)*, David A. Van Dyk and Max Welling (Eds.). JMLR.org, 169–176. http://proceedings.mlr.press/v5/goldberg09a.html

[15] T. Gournelos, V. Kotinas, and S. Poulos. 2020. Fitting a Gaussian mixture model to bivariate distributions of monthly river flows and suspended sediments. *Journal of Hydrology* 590 (2020), 125166. https://doi.org/10.1016/j.jhydrol.2020.125166

[16] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory F. Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. 2017. Deep Learning Scaling is Predictable, Empirically. *CoRR* abs/1712.00409 (2017). arXiv:1712.00409 http://arxiv.org/abs/1712.00409

[17] H. J. Scudder III. 1965. Probability of error of some adaptive pattern-recognition machines. *IEEE Trans. Inf. Theory* 11, 3 (1965), 363–371. https://doi.org/10.1109/TIT.1965.1053799

[18] Jiwon Kim, Youngjo Min, Daehwan Kim, Gyuseong Lee, Junyoung Seo, Kwangrok Ryoo, and Seungryong Kim. 2022. ConMatch: Semi-supervised Learning with Confidence-Guided Consistency Regularization. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXX (Lecture Notes in Computer Science, Vol. 13690)*, Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer, 674–690. https://doi.org/10.1007/978-3-031-20056-4_39

[19] Samuli Laine and Timo Aila. 2017. Temporal Ensembling for Semi-Supervised Learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=BJ6oOfqge

[20] Tien-Duy B. Le, David Lo, Claire Le Goues, and Lars Grunske. 2016. A learning-to-rank based fault localization approach using likely invariants. In *Proceedings of the 25th International Symposium on Software Testing and Analysis, ISSTA 2016, Saarbrücken, Germany, July 18-20, 2016*, Andreas Zeller and Abhik Roychoudhury (Eds.). ACM, 177–188. https://doi.org/10.1145/2931037.2931049

[21] Dong-Hyun Lee et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, Vol. 3. Atlanta, 896.

[22] Chun Li, Hui Li, Zhong Li, Minxue Pan, and Xuandong Li. 2025. Enhancing Fault Localization in Industrial Software Systems via Contrastive Learning . In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, Los Alamitos, CA, USA, 101–113. https://doi.org/10.1109/ICSE55347.2025.00009

[23] Junnan Li, Richard Socher, and Steven C. H. Hoi. 2020. DivideMix: Learning with Noisy Labels as Semi-supervised Learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=HJgExaVtwr

[24] Xia Li, Wei Li, Yuqun Zhang, and Lingming Zhang. 2019. DeepFL: integrating multiple fault diagnosis dimensions for deep fault localization. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2019, Beijing, China, July 15-19, 2019*, Dongmei Zhang and Anders Møller (Eds.). ACM, 169–180. https://doi.org/10.1145/3293882.3330574

[25] Xi Li, David J. Miller, Zhen Xiang, and George Kesidis. 2024. BIC-Based Mixture Model Defense Against Data Poisoning Attacks on Classifiers: A Comprehensive Study. *IEEE Transactions on Knowledge and Data Engineering* 36, 8 (2024), 3697–3711. https://doi.org/10.1109/TKDE.2024.3365548

[26] Xia Li and Lingming Zhang. 2017. Transforming programs and tests in tandem for fault localization. *Proc. ACM Program. Lang.* 1, OOPSLA (2017), 92:1–92:30. https://doi.org/10.1145/3133916

[27] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1511.05493

[28] Yi Li, Shaohua Wang, and Tien N. Nguyen. 2021. Fault Localization with Code Coverage Representation Learning. In *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*. IEEE, 661–673. https://doi.org/10.1109/ICSE43902.2021.00067

[29] Yi Li, Shaohua Wang, and Tien N. Nguyen. 2022. Fault localization to detect co-change fixing locations. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*, Abhik Roychoudhury, Cristian Cadar, and Miryung Kim (Eds.). ACM, 659–671. https://doi.org/10.1145/3540250.3549137

[30] Yiling Lou, Qihao Zhu, Jinhao Dong, Xia Li, Zeyu Sun, Dan Hao, Lu Zhang, and Lingming Zhang. 2021. Boosting coverage-based fault localization via graph-based representation learning. In *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021*, Diomidis Spinellis, Georgios Gousios, Marsha Chechik, and Massimiliano Di Penta (Eds.). ACM, 664–676. https://doi.org/10.1145/3468264.3468580

[31] Xiangxin Meng, Xu Wang, Hongyu Zhang, Hailong Sun, and Xudong Liu. 2022. Improving Fault Localization and Program Repair with Deep Semantic Features and Transferred Knowledge. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*. ACM, 1169–1180. https://doi.org/10.1145/3510003.3510147

[32] Siqi Miao, Mia Liu, and Pan Li. 2022. Interpretable and Generalizable Graph Learning via Stochastic Attention Mechanism. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*

(Proceedings of Machine Learning Research, Vol. 162), Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (Eds.). PMLR, 15524–15543. https://proceedings.mlr.press/v162/miao22a.html

[33] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2019. Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *IEEE Trans. Pattern Anal. Mach. Intell.* 41, 8 (2019), 1979–1993. https://doi.org/10.1109/TPAMI.2018.2858821

[34] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2019. Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *IEEE Trans. Pattern Anal. Mach. Intell.* 41, 8 (2019), 1979–1993. https://doi.org/10.1109/TPAMI.2018.2858821

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[36] Haim H. Permuter, Joseph M. Francos, and Ian Jermyn. 2006. A study of Gaussian mixture models of color and texture features for image classification and segmentation. *Pattern Recognit.* 39, 4 (2006), 695–706. https://doi.org/10.1016/J.PATCOG.2005.10.028

[37] Strategic Planning. 2002. The economic impacts of inadequate infrastructure for software testing. *National Institute of Standards and Technology* 1, 2002 (2002).

[38] Jie Qian, Xiaolin Ju, and Xiang Chen. 2023. GNet4FL: effective fault localization via graph convolutional neural network. *Autom. Softw. Eng.* 30, 2 (2023), 16. https://doi.org/10.1007/S10515-023-00383-Z

[39] Jie Qian, Xiaolin Ju, Xiang Chen, Hao Shen, and Yiheng Shen. 2021. AGFL: A Graph Convolutional Neural Network-Based Method for Fault Localization. In *21st IEEE International Conference on Software Quality, Reliability and Security, QRS 2021, Hainan, China, December 6-10, 2021.* IEEE, 672–680. https://doi.org/10.1109/QRS54544.2021.00077

[40] Md Nakhla Rafi, Dong Jae Kim, An Ran Chen, Tse-Hsun (Peter) Chen, and Shaowei Wang. 2024. Towards Better Graph Neural Network-Based Fault Localization through Enhanced Code Representation. *Proc. ACM Softw. Eng.* 1, FSE (2024), 1937–1959. https://doi.org/10.1145/3660793

[41] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. 2015. Semi-supervised Learning with Ladder Networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (Eds.). 3546–3554. https://proceedings.neurips.cc/paper/2015/hash/378a063b8fdb1db941e34f4bde584c7d-Abstract.html

[42] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. 2011. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, Lise Getoor and Tobias Scheffer (Eds.). Omnipress, 833–840. https://icml.cc/2011/papers/455_icmlpaper.pdf

[43] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. 2016. Regularization With Stochastic Transformations and Perturbations for Deep Semi-Supervised Learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (Eds.). 1163–1171. https://proceedings.neurips.cc/paper/2016/hash/30ef30b64204a3088a26bc2e6ecf7602-Abstract.html

[44] Razieh Sheikhpour, Mehdi Agha Sarram, Sajjad Gharaghani, and Mohammad Ali Zare Chahooki. 2017. A Survey on semi-supervised feature selection methods. *Pattern Recognit.* 64 (2017), 141–158. https://doi.org/10.1016/J.PATCOG.2016.11.003

[45] Zhengxiang Shi, Francesco Tonolini, Nikolaos Aletras, Emine Yilmaz, Gabriella Kazai, and Yunlong Jiao. 2023. Rethinking Semi-supervised Learning with Language Models. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, 5614–5634. https://doi.org/10.18653/V1/2023.FINDINGS-ACL.347

[46] Jeongju Sohn and Shin Yoo. 2017. FLUCCS: using code and change metrics to improve fault localization. In *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis, Santa Barbara, CA, USA, July 10 - 14, 2017*, Tevfik Bultan and Koushik Sen (Eds.). ACM, 273–283. https://doi.org/10.1145/3092703.3092717

[47] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. 2020. FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/06964dce9addb1c5cb5d6e3d9838f733-Abstract.html

[48] Petre Stoica and Yngve Selén. 2004. Model-order selection: a review of information criterion rules. *IEEE Signal Process. Mag.* 21, 4 (2004), 36–47. https://doi.org/10.1109/MSP.2004.1311138

[49] Jesper E. van Engelen and Holger H. Hoos. 2020. A survey on semi-supervised learning. *Mach. Learn.* 109, 2 (2020), 373–440. https://doi.org/10.1007/S10994-019-05855-6

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[51] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph Attention Networks. *CoRR* abs/1710.10903 (2017). arXiv:1710.10903 http://arxiv.org/abs/1710.10903

[52] Iris Vessey. 1985. Expertise in Debugging Computer Programs: A Process Analysis. *Int. J. Man Mach. Stud.* 23, 5 (1985), 459–494. https://doi.org/10.1016/S0020-7373(85)80054-7

[53] Wei Wang and Zhi-Hua Zhou. 2010. A New Analysis of Co-Training. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, Johannes Fürnkranz and Thorsten Joachims (Eds.). Omnipress, 1135–1142. https://icml.cc/Conferences/2010/papers/275.pdf

[54] Xiao Wang, Daisuke Kihara, Jiebo Luo, and Guo-Jun Qi. 2021. EnAET: A Self-Trained Framework for Semi-Supervised and Supervised Learning With Ensemble Transformations. *IEEE Trans. Image Process.* 30 (2021), 1639–1647. https://doi.org/10.1109/TIP.2020.3044220

[55] Yidong Wang, Hao Chen, Qiang Heng, Wenxin Hou, Yue Fan, Zhen Wu, Jindong Wang, Marios Savvides, Takahiro Shinozaki, Bhiksha Raj, Bernt Schiele, and Xing Xie. 2023. FreeMatch: Self-adaptive Thresholding for Semi-supervised Learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net. https://openreview.net/forum?id=PDrUPTXJI_A

[56] Tong Wei, Jiang-Xin Shi, Wei-Wei Tu, and Yufeng Li. 2021. Robust Long-Tailed Learning under Label Noise. *CoRR* abs/2108.11569 (2021). arXiv:2108.11569 https://arxiv.org/abs/2108.11569

[57] Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in statistics: Methodology and distribution*. Springer, 196–202. https://doi.org/10.1007/978-1-4612-4380-9_16

[58] W. Eric Wong, Ruizhi Gao, Yihao Li, Rui Abreu, and Franz Wotawa. 2016. A Survey on Software Fault Localization. *IEEE Trans. Software Eng.* 42, 8 (2016), 707–740. https://doi.org/10.1109/TSE.2016.2521368

[59] Shumei Wu, Zheng Li, Yong Liu, Xiang Chen, and Mingyu Li. 2023. GMBFL: Optimizing Mutation-Based Fault Localization via Graph Representation. In *IEEE International Conference on Software Maintenance and Evolution, ICSME 2023, Bogotá, Colombia, October 1-6, 2023*. IEEE, 245–257. https://doi.org/10.1109/ICSME58846.2023.00033

[60] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.* 32, 1 (2021), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

[61] Qizhe Xie, Minh-Thang Luong, Eduard H. Hovy, and Quoc V. Le. 2020. Self-Training With Noisy Student Improves ImageNet Classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 10684–10695. https://doi.org/10.1109/CVPR42600.2020.01070

[62] Xiaoyuan Xie, Tsong Yueh Chen, Fei-Ching Kuo, and Baowen Xu. 2013. A theoretical analysis of the risk evaluation formulas for spectrum-based fault localization. *ACM Trans. Softw. Eng. Methodol.* 22, 4 (2013), 31:1–31:40. https://doi.org/10.1145/2522920.2522924

[63] Xiaoyuan Xie, Zicong Liu, Shuo Song, Zhenyu Chen, Jifeng Xuan, and Baowen Xu. 2016. Revisit of automatic debugging via human focus-tracking analysis. In *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*, Laura K. Dillon, Willem Visser, and Laurie A. Williams (Eds.). ACM, 808–819. https://doi.org/10.1145/2884781.2884834

[64] Jiaxi Xu, Fei Wang, and Jun Ai. 2021. Defect Prediction With Semantics and Context Features of Codes Based on Graph Representation Learning. *IEEE Trans. Reliab.* 70, 2 (2021), 613–625. https://doi.org/10.1109/TR.2020.3040191

[65] Jifeng Xuan and Martin Monperrus. 2014. Learning to Combine Multiple Ranking Metrics for Fault Localization. In *30th IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, September 29 - October 3, 2014*. IEEE Computer Society, 191–200. https://doi.org/10.1109/ICSME.2014.41

[66] David Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *33rd Annual Meeting of the Association for Computational Linguistics, 26-30 June 1995, MIT, Cambridge, Massachusetts, USA, Proceedings*, Hans Uszkoreit (Ed.). Morgan Kaufmann Publishers / ACL, 189–196. https://doi.org/10.3115/981658.981684

[67] Muhan Zeng, Yiqian Wu, Zhentao Ye, Yingfei Xiong, Xin Zhang, and Lu Zhang. 2022. Fault Localization via Efficient Probabilistic Modeling of Program Semantics. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*. ACM, 958–969. https://doi.org/10.1145/3510003.3510073

[68] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. 2021. FlexMatch: Boosting Semi-Supervised Learning with Curriculum Pseudo Labeling. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December*

6-14, 2021, virtual, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 18408–18419. https://proceedings.neurips.cc/paper/2021/hash/995693c15f439e3d189b06e89d145dd5-Abstract.html

[69] Zhuo Zhang, Yan Lei, Xiaoguang Mao, and Panpan Li. 2019. CNN-FL: An Effective Approach for Localizing Faults using Convolutional Neural Networks. In *26th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2019, Hangzhou, China, February 24-27, 2019*, Xinyu Wang, David Lo, and Emad Shihab (Eds.). IEEE, 445–455. https://doi.org/10.1109/SANER.2019.8668002

[70] Tong Zhao, Wei Jin, Yozen Liu, Yingheng Wang, Gang Liu, Stephan Günnemann, Neil Shah, and Meng Jiang. 2023. Graph Data Augmentation for Graph Machine Learning: A Survey. *IEEE Data Eng. Bull.* 46, 2 (2023), 140–165. http://sites.computer.org/debull/A23june/p140.pdf

[71] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver J. Woodford, Meng Jiang, and Neil Shah. 2021. Data Augmentation for Graph Neural Networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021.* AAAI Press, 11015–11023. https://doi.org/10.1609/AAAI.V35I12.17315

[72] Wei Zheng, Desheng Hu, and Jing Wang. 2016. Fault Localization Analysis Based on Deep Neural Network. *Mathematical Problems in Engineering* 2016, 1 (2016), 1820454. https://doi.org/10.1155/2016/1820454

[73] Yu Zhou, Juanjuan Shen, Xiaoqing Zhang, Wenhua Yang, Tingting Han, and Taolue Chen. 2022. Automatic source code summarization with graph attention networks. *J. Syst. Softw.* 188 (2022), 111257. https://doi.org/10.1016/J.JSS.2022.111257

[74] Zhi-Hua Zhou and Ming Li. 2010. Semi-supervised learning by disagreement. *Knowl. Inf. Syst.* 24, 3 (2010), 415–439. https://doi.org/10.1007/S10115-009-0209-Z

[75] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph Contrastive Learning with Adaptive Augmentation. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (Eds.). ACM / IW3C2, 2069–2080. https://doi.org/10.1145/3442381.3449802

[76] Daming Zou, Jingjing Liang, Yingfei Xiong, Michael D. Ernst, and Lu Zhang. 2021. An Empirical Study of Fault Localization Families and Their Combinations. *IEEE Trans. Software Eng.* 47, 2 (2021), 332–347. https://doi.org/10.1109/TSE.2019.2892102

Received 2024-09-13; accepted 2025-01-14