Enhancing Deep Learning Vulnerability Detection through Imbalance Loss Functions: An Empirical Study

Yanzhong He
School of Computer Science and Artificial Intelligence,
Hubei Key Laboratory of Transportation Internet of Things,
Wuhan University of Technology,
Wuhan, China
heyanzhong@whut.edu.cn

Guancheng Lin
School of Cyber Science and Engineering,
Wuhan University,
Wuhan, China
guanchenglin@whu.edu.cn

Xiaoxue Ma*
Jacky Wai Keung
Department of Computer Science,
City University of Hong Kong,
Hong Kong, China
xiaoxuema3-c@my.cityu.edu.hk,
jacky.keung@cityu.edu.hk

Cheng Tan
Wenhua Hu
Fuyang Li*
School of Computer Science and Artificial Intelligence,
Hubei Key Laboratory of Transportation Internet of Things,
Wuhan University of Technology,
Wuhan, China
{cheng_tan,whu10,fyli}@whut.edu.cn

ABSTRACT

Software Vulnerability Detection (VD) is crucial in software engineering, and Deep Learning (DL) has demonstrated effective in this domain. However, the class imbalance issue, where non-vulnerable code snippets vastly outnumber vulnerable ones, hinders the performance of DL-based Vulnerability Detection (DLVD) models. Recent studies have explored data resampling methods to address this, but these methods often lead to data distribution alterations, resulting in information loss, model overfitting, and reduced interpretability. Imbalance loss functions have thus emerged as viable alternatives. To comprehensively evaluate the effectiveness of imbalance loss functions in DLVD, we investigate six imbalance loss functions and Cross-Entropy Loss (the default for LineVul and ReVeal models) on two DLVD models across three public VD datasets, using three evaluation metrics and the Scott-Knott Effect Size Difference test. Our findings provide valuable insights into selecting loss functions and data resampling methods in DLVD. First, the DLVD model LineVul outperforms ReVeal across all datasets. Second, Label Distribution-Aware Margin loss and Random Under-Sampling generally yield the best Precision and Recall, respectively. Third, to avoid information loss and maintain interpretability, we recommend Logit Adjustment Loss (LALoss) due to its high Recall and superior F1 metric performance. Based on these findings, we suggest employing LineVul with LALoss for VD, as it detects more vulnerable code snippets (higher Recall) while providing comprehensive performance (higher F1).

* Xiaoxue Ma and Fuyang Li are the corresponding authors.

Internetware 2024, July 24–26, 2024, Macau, China

ACM ISBN 979-8-4007-0705-6/24/07
https://doi.org/10.1145/3671016.3671379

CCS CONCEPTS

• **Security and privacy → Software security engineering**; • **Software and its engineering**;

KEYWORDS

Vulnerability detection, deep learning, imbalance loss functions, data resampling

ACM Reference Format:
Yanzhong He, Guancheng Lin, Xiaoxue Ma*, Jacky Wai Keung, Cheng Tan, Wenhua Hu, and Fuyang Li*. 2024. Enhancing Deep Learning Vulnerability Detection through Imbalance Loss Functions: An Empirical Study. In *15th Asia-Pacific Symposium on Internetware (Internetware 2024), July 24–26, 2024, Macau, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3671016.3671379

1 INTRODUCTION

Software vulnerability detection is a pivotal task within the domains of software engineering and information security [5–7, 9, 24, 28, 29, 49, 50, 52, 54]. Recently, with the widespread application of deep learning technology in software engineering [51–53], researchers have extensively applied Deep Learning (DL) techniques to autonomously acquire and extract vulnerability features from code snippets, showcasing noteworthy success in Vulnerability Detection (VD) [5, 9, 14, 15, 20–22, 32, 37, 46, 48–50]. However, a significant and practical challenge that cannot be overlooked in VD is class imbalance (i.e., the count of non-vulnerable code snippets in the training set significantly exceeds that of vulnerable code snippets). Chakraborty et al. [5] found that Deep Learning Vulnerability Detection (DLVD) models tend to favor the majority class (i.e., the non-vulnerable class) on imbalanced datasets, leading to poor Recall. Specifically, a low Recall indicates that DLVD models may overlook or inadequately identify numerous vulnerable code snippets. To alleviate this issue, data resampling methods are commonly employed in software engineering [1, 19, 34, 41]. Yang et al. [41]

extensively investigated the effects of four data resampling methods (i.e., Random Under-Sampling (RUS), Random Over-Sampling (ROS), Synthetic Minority Oversampling Technique (SMOTE), and One-Side Selection (OSS)) on DLVD. They found that ROS enhances both the Precision and Recall values of DLVD models, while RUS specifically improves the Recall value.

However, data resampling methods have notable limitations [1, 16, 27, 33–35]. They can alter the dataset's distribution: undersampling may result in the loss of valuable information while oversampling can introduce overfitting [1, 27, 35]. In addition, resampling can reduce model interpretability by making the training dataset unrepresentative of the original data [16, 27, 33–35]. Consequently, in DL-based software engineering, prior studies have utilized imbalance loss functions to address class imbalance challenges [13, 40]. These loss functions adjust the model's misclassification costs across different classes, focusing on the minority class without altering the dataset's overall distribution [36]. Despite their widespread adoption, no research has systematically and comprehensively examined loss functions specifically designed to tackle class imbalance in DLVD.

Therefore, we aim to assess the impact of imbalance loss functions on the performance of existing DLVD approaches. To achieve this objective, we conduct an extensive empirical study by employing six imbalance loss functions (Logit Adjustment Loss (LALoss) [26], Label-Distribution-Aware Margin Loss (LDAMLoss) [3], Gradient Harmonized Margin Loss (GHMLoss) [18], Focal Loss (FocalLoss) [23], Class Balanced Cross-Entropy Loss (CBCELoss) [10], and Class Balanced Focal Loss (CBFocalLoss) [10]) and Cross-Entropy Loss (CELoss) [31] to two DLVD approaches (LineVul [14] and ReVeal [5]) across three publicly available vulnerability datasets. Moreover, we analyze the performance disparities between imbalance loss functions and conventional data resampling methods (RUS and ROS). To summarize, our study is structured by the following Research Questions (RQs):

(1) Which loss function performs best under the two DLVD models? We assess the influence of utilizing six imbalance loss functions and CELoss on two DLVD models across three datasets. **Findings:** LineVul consistently outperforms ReVeal across all three datasets. Using the LineVul model for VD with LDAMLoss optimizes high Precision, while LALoss achieves high Recall. Overall, LALoss is recommended for overall performance in terms of the F1 metric.

(2) Do the loss functions yield superior results compared to the resampling methods? We compare the performance differences between imbalance loss functions and data resampling methods on two DLVD models. **Findings:** LDAMLoss is recommended for high Precision, and RUS is recommended for high Recall. However, it's important to note that RUS can lead to information loss and reduced model interpretability [16, 27, 33–35]. Consequently, we recommend LALoss, as it not only achieves high Recall but also delivers the best performance on the F1 metric.

The remainder of this paper is organized as follows: Section 2 briefly describes the DLVD models, loss functions, and data resampling methods studied. Section 3 and 4 present the experimental setup and results. Section 5 discusses the implications of our findings and the threats of validity. Section 6 presents related work in DLVD and empirical studies on imbalance learning in software engineering tasks. Finally, Section 7 concludes the study.

2 PRELIMINARIES

DLVD uses DL technology to automatically discover and identify vulnerable code snippets in software. A code snippet can be represented as $C_i = (\boldsymbol{x}_i, y_i)$, where $\boldsymbol{x}_i$ is the embedding features vector of the i-th code snippet, and y_i represents the class label (0 for non-vulnerable and 1 for vulnerable) of the i-th code snippet. A VD dataset can be represented as $D = \{C_i = (\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, where N is the total number of code snippets in D. The objective of VD is to train a binary classification model from D to predict whether new code snippets contain vulnerabilities. Given an input code snippet $\boldsymbol{x}$, the DLVD model processes it to yield a predicted output vector $z = [z_0, z_1]^\top$, where z_0 and z_1 represent the model's scores for the two different classes. The loss function treats each class as mutually exclusive, calculating the vulnerable probability distribution across the two classes as $p = \frac{e^{z_1}}{e^{z_0}+e^{z_1}}$.

2.1 DLVD Models

Given the rapidly evolving landscape of DLVD research, it is challenging to single out one or even a few techniques to represent the entire community. In our experiments, we select LineVul [14] as the token-based representative approach and ReVeal [5] as the graph-based representative approach. These two approaches serve as common baselines for evaluating the generalizability of proposed methods within the VD community [39, 41, 46]. LineVul converts code snippets into tokens and generates embedding vectors, employing a BERT architecture with a stack of 12 Transformer encoder blocks for VD. ReVeal transforms code snippets into code property graphs and utilizes a graph-gated neural network to detect vulnerable code snippets using feature vectors of all nodes and edges as input. The **Cross-Entropy Loss (CELoss)** [31] serves as the loss function for both LineVul and ReVeal in prior research [5, 14], as well as for other DLVD models [46, 48, 56]. The standard CELoss is formulated as follows:

$$\mathcal{L}_{\text{CE}} = \sum_{i=1}^{N} -y_i log(p_i) - (1 - y_i) log(1 - p_i) \tag{1}$$

where p_i reflects the predicted probability that the i-th code snippet is vulnerable. CELoss treats each training code snippet equally during optimization. However, it tends to prioritize a greater proportion of non-vulnerable code snippets (i.e., the majority class), which can negatively impact the model's ability to accurately identify vulnerable code snippets (i.e., the minority class), especially when faced with high class imbalance.

2.2 Imbalance Loss Functions

To address this issue of class imbalance in vulnerability datasets, we investigate the impact of six widely used imbalance loss functions and CELoss for DLVD. We classify these imbalance loss functions into two groups based on their optimization approaches for CELoss: (1) adjusting probability (i.e., Logit Adjustment Loss and Label-Distribution-Aware Margin Loss) and (2) adjusting weight (i.e., Gradient Harmonized Margin Loss, Focal Loss, Class Balanced Cross-Entropy Loss, and Class Balanced Focal Loss).

(1) **Logit Adjustment Loss (LALoss)** [26]. It introduces label-dependent offsets by incorporating class priors, aiding in the adjustment of each logit to better align with the training samples, particularly in handling imbalanced data. Consequently, the vulnerability probability $p = \frac{e^{z_1}}{e^{z_0}+e^{z_1}}$ is refined to $p_{\text{LA}} = \frac{e^{z_1+\tau\cdot\log\pi_1}}{e^{z_0+\tau\cdot\log\pi_0}+e^{z_1+\tau\cdot\log\pi_1}}$. The refined standard CELoss is formulated as follows:

$$\mathcal{L}_{\text{LA}} = \sum_{i=1}^{N} -y_i log(p_{\text{LA}_i}) - (1-y_i) log(1-p_{\text{LA}_i}) \tag{2}$$

where p_{LA_i} represents the predicted probability that the i-th code snippet is vulnerable, π signifies estimates of class priors, and τ is the temperature parameter.

(2) **Label-Distribution-Aware Margin Loss (LDAMLoss)** [3]. It introduces class-dependent margin factors determined by the training label frequencies for each class to tackle the challenge of class imbalance. As a result, the vulnerability probability $p = \frac{e^{z_1}}{e^{z_0}+e^{z_1}}$ is refined to $p_{\text{LDAM}} = \frac{e^{z_1-\Delta_1}}{e^{z_0-\Delta_0}+e^{z_1-\Delta_1}}$. The refined standard CELoss is formulated as follows:

$$\mathcal{L}_{\text{LDAM}} = \sum_{i=1}^{N} -y_i log(p_{\text{LDAM}_i}) - (1-y_i) log(1-p_{\text{LDAM}_i}) \tag{3}$$

where p_{LDAM_i} represents the predicted probability that the i-th code snippet is vulnerable, and Δ_y denotes the class-dependent margin for the true class.

(3) **Gradient Harmonized Margin Loss (GHMLoss)** [18]. It pays more attention to the distribution of sample gradients in order to better balance the loss contributions between difficult and easy code snippets. In VD, an easy (or difficult) code snippet is one that the model has a high (or low) probability of correctly classifying. The weight of code snippets with high gradient density will be reduced, and the weight of code snippets with small density will be increased. The standard CELoss formula using gradient density weights is as follows:

$$\mathcal{L}_{\text{GHM}} = \sum_{i}^{N} -\alpha_{GD_y} y_i log(p_i) - \alpha_{GD_y}(1-y_i) log(1-p_i) \tag{4}$$

where α_{GD_y} is a gradient density weighting factor that can be used to assign different importance to different classes according to the gradient.

(4) **Focal Loss (FocalLoss)** [23]. It adjusts CELoss through the incorporation of a class-dependent weighting factor (α) and a focusing parameter (γ):

$$\mathcal{L}_{\text{FL}} = \sum_{i}^{N} -\alpha_y (1-p_i)^\gamma y_i log(p_i) - \alpha_y p_i^\gamma (1-y_i) log(1-p_i) \tag{5}$$

where p_i^γ ($\gamma \in [0,5]$) serves to mitigate the impact of easily classified non-vulnerable code snippets on the overall loss. a_y is a weight factor that can be used to assign different importance to different classes. It is usually set higher for the minority class (vulnerable code snippets) to account for the class imbalance.

(5) **Class Balanced Cross-Entropy Loss (CBCELoss)** [10]. A weight factor $\frac{1}{E_{n_y}}$ that is inversely proportional to the number of effective code snippets (i.e., the expected volume of the dataset) is introduced to address the class imbalance problem:

$$E_{n_y} = \sum_{j=1}^{n} \beta^{j-1} = \frac{1-\beta^n}{1-\beta} \tag{6}$$

where n_y represents the number of code snippets belonging to one class y, and $\beta \in [0,1]$ controls how fast E_{n_y} grows as n_y increases. This means that the j^{th} data contributes β^{j-1} to the effective number. As j increases, the information contained in j^{th} code snippet may already exist in the previous code snippet, leading to a gradual decrease in their contribution to the effective number. After adding the weight factor $\frac{1}{E_{n_y}}$ in the standard CELoss, the following CBCELoss is obtained as follows:

$$\mathcal{L}_{\text{CBCE}} = \sum_{i=1}^{N} -\frac{1}{E_{n_y}} y_i \log(p_i) - \frac{1}{E_{n_y}} (1-y_i) \log(1-p_i) \tag{7}$$

(6) **Class Balanced Focal Loss (CBFocalLoss)** [10]. It combines the ideas of class balance and FocalLoss by introducing the weight factor of effective samples and focal factor while paying attention to the difficulty of the code snippets and the balance of classes. Similarly, the weight factor $\frac{1}{E_{n_y}}$ serves as a replacement for α_y in FocalLoss. After replacing the weight factors in the standard FocalLoss, the following CBFocalLoss is obtained as follows:

$$\mathcal{L}_{\text{CBFL}} = \sum_{i}^{N} -\frac{1}{E_{n_y}} (1-p_i)^\gamma y_i log(p_i) - \frac{1}{E_{n_y}} p_i^\gamma (1-y_i) log(1-p_i) \tag{8}$$

2.3 Data Resampling

We consider two key criteria when selecting the compared data resampling methods with the imbalance loss function. First, we want to ensure that the chosen methods cover both oversampling and undersampling techniques. Second, according to Yang's experimental results [41], RUS and ROS perform best in DLVD. Therefore, we choose RUS and ROS as the compared methods in our study. We use the data resampling method on the training dataset to obtain the balanced training dataset and apply the standard cross-entropy loss function for model training. To maintain consistency with previous studies [19, 41], we set the default optimal vulnerable ratio to 0.5, resulting in an equal number of vulnerable and non-vulnerable code snippets in the balanced datasets.

(1) **Random Under-Sampling (RUS)** randomly deletes m non-vulnerable code snippets from the original vulnerability dataset (m is calculated based on the default optimal vulnerable ratio).

(2) **Random Over-Sampling (ROS)** randomly copies n vulnerable code snippets from the original set of vulnerable code snippets (n is calculated based on the default optimal vulnerable ratio). Subsequently, these duplicated code snippets are added to the original vulnerability dataset, resulting in a balanced dataset.

However, it is pertinent to acknowledge that data resampling methods exhibit certain limitations [1, 16, 27, 33–35]. Among these, a notable concern is their propensity to alter the distribution of the dataset. For instance, undersampling methods may result in the loss of valuable information, whereas oversampling methods may introduce overfitting problems [1, 27, 35]. Furthermore, these methods have the potential to diminish the interpretability of models [16, 27, 33–35]. Specifically, they can introduce bias into learned concepts, causing concept drift when the class distributions of the

training and testing datasets differ [16, 17, 33, 35]. As a result, resampling the training dataset may make it unrepresentative of the original data, thereby affecting the interpretation of VD models. Consequently, it becomes imperative to consider the trade-offs between model performance and interpretability when employing data resampling techniques.

3 EXPERIMENTAL DESIGN

Datasets. According to the source of vulnerability label acquisition of code snippets, open source datasets in the field of VD can be divided into four types of datasets: developer-provided, security vendor-provided, tool-created, and synthetically created [5, 9]. Due to developers' careful manual annotation, Devign [56] and ReVeal$_D$ [5] have higher data quality assurance. Therefore, we prefer these two datasets as core experimental datasets. To enhance the generalizability of our study and diversify our data sources, we also use the synthetically created Juliet dataset [2]. Research by Roland et al. [9] identified duplicate and conflicting code snippets in these three commonly used VD datasets, which can impact model training and distort benchmark performance. To address these issues, we use the cleaned versions of Devign and Juliet provided by Roland et al. [9]. However, Roland et al. [9] were unable to clean the ReVeal$_D$ dataset due to the lack of appropriate metadata. Therefore, in our experiments, we use the original ReVeal$_D$ dataset. A comprehensive summary of the dataset's statistical information is presented in Table 1.

Table 1: The statistical information of our experimental datasets.

Dataset	# Code Snippets	Source of Annotation	Vulnerable Ratio
Devign	24,491	developer-provided	45.72%
ReVeal$_D$	22,734	developer-provided	9.85%
Juliet	38,234	synthesized-created	40.50%

Evaluation Metrics. Following previous VD researches [5, 14, 30, 37–39, 41, 46, 48], we use $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$, and $F1\text{-score} = 2\times\frac{Precision\times Recall}{Precision+Recall}$ as our evaluation metrics. These metrics are employed to comprehensively assess the performance of the DLVD models. In the given equations, TP is the count of correctly identified vulnerable code snippets, FN is the count of vulnerable ones mistakenly predicted as non-vulnerable, FP is the count of non-vulnerable code snippets incorrectly predicted as vulnerable, and TN is the count of correctly predicted non-vulnerable ones.

Experimental procedure. Figure 1 shows the framework of our empirical study. We use the same data splitting method with the study of LineVul [14] and ReVeal [5], specifically random dividing the entire dataset into 80% for training, 10% for validation, and 10% for testing. The training set is first used to train the LineVul and ReVeal models using six imbalance loss functions and CELoss as the loss function. After that, the training set is applied RUS and ROS to obtain the balanced dataset. The balanced dataset is used to train the LineVul and ReVeal models. The validation set helps the models update optimal parameters during the above training process. The test set evaluates the performance of the trained

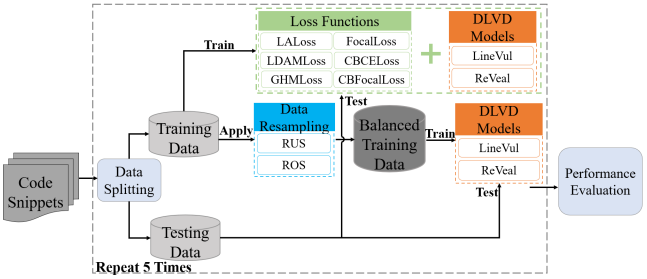


Figure 1: The framework of our study.

models. This process is repeated five times. So the five results are obtained on one dataset for each loss function and data resampling method. We use the same hyperparameter settings described in the papers of LineVul [14] and ReVeal [5], including learning rate, batch size, and the number of training epochs. Based on previous research [3, 10, 18, 23, 26], we notice that certain parameter settings of the loss function achieved the best results, so we decide to adopt these parameter values. Specifically, we set τ in LALoss to 1, set γ in FocalLoss, and CBFocalLoss to 2. Also, we configure π in LALoss, Δ in LDAMLoss, α_{GD} in GHMLoss, α in FocalLoss, as well as $\frac{1}{E_{n_y}}$ in CBCELoss and CBFocalLoss based on the original class imbalance proportions present in the vulnerability dataset. In our preliminary experiments, we adjust some parameter values, and the results confirm that the parameters we currently select can achieve the best results. During the training phase, we employ backpropagation in conjunction with the AdamW optimizer, which is utilized to update the model and minimize the loss function. Our experiments are conducted on NVIDIA RTX 3090 24GB GPU, Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz CPU.

Statistic Test. The Scott-Knott Effect Size Difference (ESD) test [34] is a multiple comparison method that employs hierarchical clustering to categorize loss functions and data resampling methods into distinct groups with significant differences at the significance level of 0.05. There are no significant differences between loss functions and data resampling methods placed in the same group, but there are significant differences between loss functions and data resampling methods placed in different groups.

4 RESULTS

4.1 RQ1: Which loss function performs best under the two DLVD models?

To address RQ1, we first evaluate the impact of employing six widely used imbalance loss functions and CELoss on two DLVD models (LineVul and ReVeal) across three datasets (Devign, ReVeal$_D$, and Juliet) in terms of three evaluated metrics (Precision, Recall, and F1). The significant differences among different methods are presented as rankings through the Scott-Knott ESD test in Figures 2(a)–2(c) (for LineVul) and 3(a)–3(c) (for ReVeal). Lower rankings indicate superior performance. Different rankings signify significant differences between the methods, while the same ranking indicates no significant difference between the two methods. In addition, we present the average results of five runs in Figures 4(a)–4(c) (for LineVul) and 5(a)–5(c) (for ReVeal). In these figures, larger values and darker colors in table cells indicate better results.

Figure 2: The Scott-Knott ESD ranking of loss functions and data resampling methods for all datasets based on LineVul model

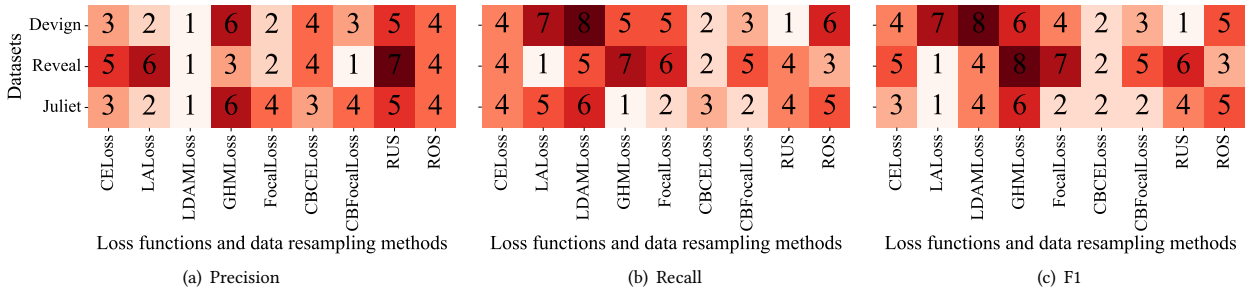
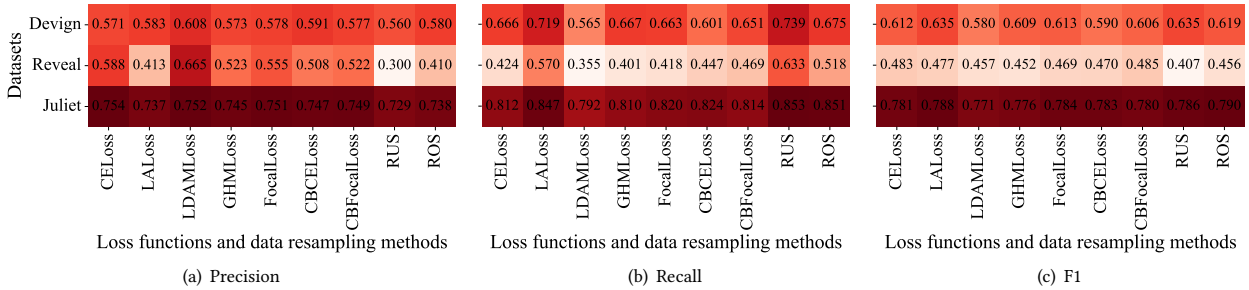


Figure 3: The Scott-Knott ESD ranking of loss functions and data resampling methods for all datasets based on ReVeal model

Result: (1) When utilizing LineVul as the DLVD model, LDAMLoss demonstrates the highest Scott-Knott ESD ranking for the Precision metric across all three datasets, as shown in Figure 2(a). Additionally, the average Precision values in Figure 4(a) reveal LDAMLoss achieving 0.608 on Devign, 0.665 on ReVeal$_D$, and 0.752 on Juliet. These findings denote improvements ranging from 6.5% to 13.1% compared to CELoss. In terms of the Recall metric, Figure 2(b) indicates that LALoss outperforms other methods and ranks highest across all three datasets. Similarly, Figure 4(b) demonstrates that LALoss achieves an average Recall of 0.719 on Devign, 0.570 on ReVeal$_D$, 0.789 on Bigvul, and 0.847 on Juliet. Compared with CELoss, LALoss shows improvements of 4.3%–34.4% across the three datasets. In terms of F1, Figure 2(c) indicates that on two nearly balanced datasets (Devign and Juliet), LALoss achieves the best ranking. However, on the extremely imbalanced dataset (ReVeal$_D$), CELoss and CBFocalLoss exhibit the best rankings, followed by LALoss. Furthermore, the average F1, presented in Figure 4(c), shows that LALoss achieves 0.635 on Devign and 0.788 on Juliet, with improvements of 3.8% and 0.9% compared to CELoss, respectively. On the extremely imbalanced dataset (ReVeal$_D$), CBFocalLoss achieves the best scores, with the values of 0.485. This value represents an improvement of 0.4% compared to CELoss. When considering the combined average results across all three datasets, LALoss achieves the best F1 score of 0.633, reflecting a 1.5% improvement over CBFocalLoss.

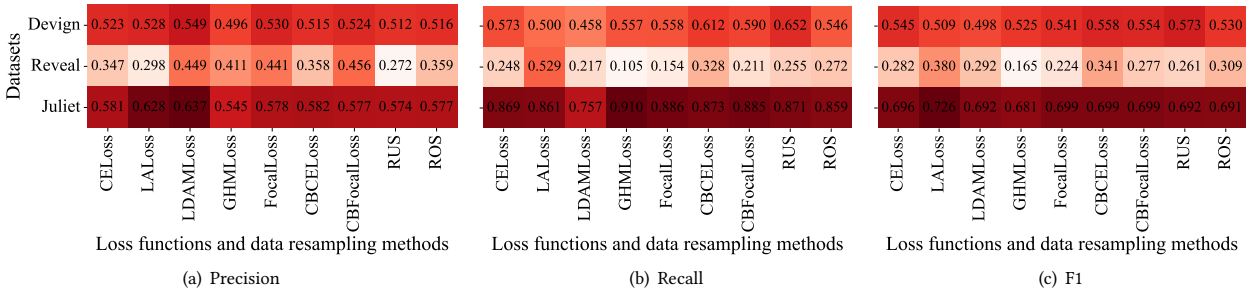
(2) When employing ReVeal as the DLVD model, Figure 3(a) shows that LDAMLoss has the highest Scott-Knott ESD ranking of three datasets on Precision. Moreover, we find the average data in Figure 5(a): on Devign, ReVeal$_D$, and Juliet, LDAMLoss respectively achieves 0.549, 0.449, and 0.637. The improvements of LDAMLoss compared to CELoss are 5.0% (Devign), 29.4% (ReVeal$_D$), and 9.6% (Juliet). In terms of Recall, Figure 3(b) indicates that CBCELoss achieves the best ranking on Devign. On the extremely imbalanced dataset (ReVeal$_D$), LALoss ranks the best. On Juliet, GHMLoss achieves the highest ranking. Furthermore, the average values in Figure 5(b) present that on Devign, CBCELoss achieves the optimal value of 0.612, marking a 6.8% improvement over CELoss. On the highly imbalanced dataset (ReVeal$_D$), LALoss achieves the best value of 0.529, representing an improvement of 113.3% compared to CELoss. On Juliet, GHMLoss attains the best Recall value of 0.910, demonstrating a 4.7% improvement over CELoss. Regarding the F1 comprehensive metric, Figure 3(c) illustrates that CBCELoss exhibits the best ranking performance on Devign. Conversely, LALoss outperforms on the ReVeal$_D$ and Juliet datasets. Analyzing the average values presented in Figure 5(c), CBCELoss achieves the highest F1 value of 0.558 on Devign, marking a 2.4% improvement over CELoss. On ReVeal$_D$ and Juliet, LALoss attains the top F1 values of 0.380 and 0.726, respectively. These values represent substantial improvements of 34.8% and 4.3% compared to CELoss on the respective datasets. While considering the combined average results across all three datasets, LALoss performs best and achieves an F1 score of 0.538, reflecting a 6.0% improvement over CELoss.

Summary: As depicted in Figures 4(a)–4(c), as well as Figures 5(a)–5(c), LineVul consistently outperforms ReVeal across three datasets. Consequently, we recommend utilizing the LineVul model for VD. For selecting the appropriate loss function, LDAMLoss is preferable when prioritizing high Precision, while LALoss is more suitable for achieving high Recall. For comprehensive performance (high F1), LALoss is recommended. It is noteworthy that LineVul when paired with certain loss functions, can exhibit superior performance compared to using CELoss in all evaluation metrics. This suggests that CELoss may not always be the optimal choice as the loss function for DLVD models.



(a) Precision (b) Recall (c) F1

Figure 4: The average value of these loss functions and data resampling methods for all datasets based on LineVul model



(a) Precision (b) Recall (c) F1

Figure 5: The average value of the loss functions and data resampling methods for all datasets based on ReVeal model

Answer to RQ1: LineVul outperforms ReVeal across all datasets, with LineVul with LALoss showing the best performance under both Recall and F1 metrics. Hence, LineVul with LALoss is recommended in DLVD.

4.2 RQ2: Do the loss functions yield superior results compared to the resampling methods?

To address RQ2, we compare the performance statistical differences between the imbalance loss functions and the data resampling methods (ROS and RUS) on two DLVD models.

Result: (1) When utilizing LineVul as the DLVD model, Figure 2(a) indicates that LDAMLoss achieves the highest Scott-Knott ESD ranking among all methods for the Precision metric across three datasets. Approximately 71.4% of the loss functions (i.e., LALoss, LDAMLoss, FocalLoss, CBCELoss, and CBFocalLoss) rank higher than the two data resampling methods on the three datasets. Moreover, the average Precision values in Figure 4(a) reveal that LDAMLoss and CBCELoss outperform data resampling methods numerically across the three datasets, showing improvements of 1.9%–121.7%. Specifically, the optimal loss function method LDAMLoss achieves 0.608 on Devign, 0.665 on ReVeal$_D$, and 0.752 on Juliet. In comparison, the optimal data resampling method ROS achieves Precision scores of 0.580 on Devign, 0.410 on ReVeal$_D$, and 0.738 on Juliet. LDAMLoss demonstrates a significant improvement of 1.9% to 62.2% compared to ROS. In terms of the Recall metric, Figure 2(b) indicates that LALoss has the highest ranking among the loss functions, and RUS has the best ranking among all methods. No loss function consistently outperforms data resampling methods across all datasets. Figure 4(b) illustrates that on the Devign, ReVeal$_D$, and Juliet datasets, the best loss function, LALoss, achieves an average Recall of 0.719, 0.570, and 0.847, respectively, while the optimal data resampling method RUS achieves Recall values of 0.739, 0.633, and 0.853, respectively. RUS exhibits a moderate improvement of 0.7% to 11.1% compared to LALoss. In terms of the comprehensive metric F1, LALoss emerges as the optimal loss function method, ranking better than data resampling methods across Devign and ReVeal$_D$ datasets, as depicted in Figure 2(c). Similarly, the average F1 presented in Figure 4(c) shows that the optimal loss function method LALoss achieves the best scores, with respective values of 0.635 (Devign), 0.477 (ReVeal$_D$), and 0.788 (Juliet). Compared to ROS, LALoss improves by 0.3%–4.6%. On the imbalanced dataset (ReVeal$_D$), each loss function (except GHMLoss) outperforms the data resampling methods by 1.1%–21.7%. Under close-to-balanced datasets (Devign and Juliet), LALoss improves by 0%–2.6% compared to the two data resampling methods.

(2) When utilizing ReVeal as the DLVD model, LDAMLoss exhibits the highest Scott-Knott ESD ranking across three datasets for Precision, as shown in Figure 3(a). About 57.1% of the loss functions (LDAMLoss, FocalLoss, CBCELoss, and CBFocalLoss) rank higher than data resampling methods across three datasets. The average values depicted in Figure 5(a) further support this trend: LDAMLoss and CBCELoss consistently outperform data resampling methods with improvements ranging from 6.4% to 65.1%. Specifically, LDAMLoss achieves Precision scores of 0.549 (Devign), 0.449 (ReVeal$_D$), and 0.637 (Juliet). In comparison, the optimal data resampling method ROS achieves 0.516 (Devign), 0.359 (ReVeal$_D$), and 0.577 (Juliet). LDAMLoss improves by 6.4% to 25.1% compared to ROS across all datasets. In terms of Recall, Figure 3(b) indicates that RUS achieves the best ranking on Devign dataset, while LALoss achieves the best ranking on ReVeal$_D$, and GHMLoss achieves the best ranking on Juliet. Examining the average values

depicted in Figure 5(b), CBCELoss, as the optimal loss function on Devign, achieves a Recall of 0.612, whereas RUS, the optimal sampling method, achieves 0.652, indicating a 6.5% improvement over CBCELoss. On ReVeal$_D$, LALoss emerges as the optimal loss function, attaining a Recall of 0.529, which significantly surpasses the best-performing sampling method ROS, achieving a Recall of 0.272. This represents a noteworthy 94.5% improvement by LALoss over ROS. Lastly, on Juliet, the optimal loss function GHMLoss achieves a Recall of 0.910, surpassing the optimal sampling method RUS, which achieves a Recall of 0.871, showing a 4.5% increase by GHMLoss. Regarding the comprehensive metric F1, Figure 3(c) illustrates that on Devign, CBCELoss achieves the best ranking of all methods; on ReVeal$_D$ and Juliet, LALoss achieves the best ranking of all methods. It is worth noting that CBCELoss outperforms data resampling methods in overall ranking on all three datasets. Analyzing the average values presented in Figure 5(c), the overall performance of CBCELoss on the three datasets is better than that of the two data resampling methods, with an improvement of 0.1%–40.0%. On Devign, RUS demonstrates superior performance compared to loss function methods. RUS achieves an optimal value of 0.573, reflecting an improvement ranging from 2.7% to 15.1% compared to loss function methods. On ReVeal$_D$, both LALoss and CBCELoss demonstrate superior performance compared to data resampling methods. Specifically, LALoss achieves an optimal value of 0.380, representing a substantial improvement of 23.0% to 45.6% over data resampling methods. Similarly, on Juliet, CELoss, LALoss, LDAMLoss, FocalLoss, CBCELoss, and CBFocalLoss outperform data resampling methods with an improvement of 0 to 5.1%. LALoss, in particular, attains the optimal value of 0.726, showcasing an enhancement of 4.9% to 5.1% compared to data resampling methods.

Summary: As depicted in Figures 4(a)–4(c), as well as Figures 5(a)–5(c), the overall performance of the imbalance loss functions (e.g., LALoss, CBCELoss, and CBFocalLoss) in Precision and F1 metrics is better than the data resampling methods. However, in terms of Recall, the data resampling methods are overall better than the imbalance loss functions. When deciding on the appropriate method, LDAMLoss is preferable for prioritizing high Precision, while RUS is more suitable for achieving high Recall. Due to the reduced interpretability caused by data resampling methods [16, 27, 33–35] (detailed in Section 2.3), we recommend LALoss as an alternative. LALoss offers superior model interpretability compared to data resampling methods, along with high Recall. In summary, we recommend employing LineVul with LALoss for DLVD.

Answer to RQ2: LDAMLoss is preferable when prioritizing high Precision, while RUS is more suitable for achieving high Recall. However, RUS has the disadvantage of losing information and reducing model interpretability. For comprehensive performance, including high F1 and model interpretability, we recommend LALoss, as it excels in both high Recall and overall F1 score.

5 DISCUSSION

5.1 Implications

The analysis of the experiments demonstrated the superiority of LDAMLoss, LALoss, and RUS over other loss functions and data resampling methods in certain cases. At the same time, the performance of LineVul is better than ReVeal on all datasets. This research carries several practical implications for practitioners.

(1) Prioritize the effective use of data information: For DLVD, software security experts often invest significant time in collecting high-quality code snippets and labeling them effectively [9]. Efficiently leveraging information from the dataset during training enhances the performance of VD models. Software engineering tasks frequently utilize imbalance loss functions or data resampling methods to alleviate the class imbalance problem [13, 40, 41]. However, data resampling may result in the loss and redundancy of some data information. For instance, when dataset information is limited, ROS may cause data redundancy and overfitting problems, thus increasing model training time and computing costs [1, 27]. On the other hand, RUS may lead to the loss of valuable data information, resulting in subpar model performance [1, 27]. In contrast, the imbalance loss function neither discards nor duplicates data information, enabling effective exploitation of the VD dataset's information. Therefore, based on our research findings, we propose combining the imbalance loss function with the LineVul model to maximize the effective utilization of VD data information.

(2) Prioritize the effectiveness of methods: When faced with information-rich VD datasets, practitioners prioritize the performance of DLVD. LALoss generally outperforms data resampling methods in terms of Precision and F1 metrics. While data resampling methods, particularly RUS, excel in achieving high Recall, they may sacrifice information and reduce model interpretability [16, 27, 33–35]. Hence, we also recommend employing the second-best method, LALoss, to achieve high Recall. Overall, we recommend LineVul prioritize LDAMLoss for high Precision, RUS and LALoss for high Recall, and LALoss for high comprehensive performance (high F1) in DLVD.

5.2 Threats of Validity

Construct validity. In order to guarantee the validity and reliability of our findings, we implement robust methodologies throughout the experimental study. In order to avoid potential bias, we adopt two representative models (LineVul and ReVeal) in the field of DLVD, as well as eight imbalanced learning methods. While we acknowledge the existence of other DLVD models and imbalanced learning methods, the purpose of our study is not to compare the effectiveness of these DLVD models. Instead, we investigate the effectiveness of imbalance loss functions in DLVD. There are other imbalance loss functions that have not been studied. In the future, we will explore the impact of other imbalance loss functions on DLVD. To minimize technical errors, we use third-party Python libraries such as imbalanced-learn [1] to implement these data resampling methods.

External validity. Our findings are based on cleaned versions of three VD datasets provided by Roland et al.: Devign and Juliet, and the unprocessed ReVeal$_D$ dataset. Although these datasets have been widely used in recent research [9, 41], it is crucial to emphasize that the generalizability of our findings to other datasets may be limited. Consequently, we intend to investigate the potential impact of other datasets on our methods in future research.

[1] *https://github.com/scikit-learn-contrib/imbalanced-learn*

Internal validity. One threat to internal validity in the DLVD model approach is associated with the hyperparameter settings during training. Hyperparameter tuning is costly for the methods we studied, which consist of millions of parameters. We use parameters recommended in previous studies. To migrate threats, we reuse the implementation published by the authors and followed the experimental setup of the DLVD model approach specified in their paper. Some parameters of imbalance loss functions are set based on previous research [3, 10, 18, 23, 26], while the parameters related to weights are configured according to the proportion of vulnerabilities in the original dataset. In future work, we will study the impact of changing the hyperparameters of the imbalanced learning method on the DLVD model.

6 RELATED WORK

6.1 Deep Learning-based VD

With the widespread application of deep learning in software engineering [8, 25, 42, 43, 47], many researchers have applied deep learning technology to the field of software vulnerability detection. Existing DLVD models can be divided into two types: token-based models and graph-based models. The former first treats code as a sequence of tokens, represented as vectors through text embedding techniques. Then these methods use different neural networks to automatically extract vulnerability features from the vectors [14, 22, 32]. For example, Russell et al. [32] utilized Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) to capture vulnerability features from code token sequences. VulDeePecker [22] and SySeVR [21] first extracted code slices from specific "interesting points" in the code (e.g., API calls, array indexing, and pointer usage), since not every line of code holds equal importance for VD. By focusing only on these code slices, the models utilized LSTM and GRU to enhance VD performance without considering the remaining code. Furthermore, Fu et al. [14] proposed LineVul, a Transformer-based line-level VD approach, which can capture the long-dependency relationship of code token sequences. Graph-based DLVD models [5, 15, 20, 37, 46, 48] first transform code snippets into various graphs and then employ graph neural networks to learn the structural feature of the code for VD. For example, Devign [56], ReVeal [5], and IVDetect [20] employed gated graph recurrent network, graph neural network, or graph convolution network to capture structural features from abstract syntax tree, control flow graph, data flow graph, code property graph, or program dependency graph. Cao et al. [4] employed a flow-sensitive graph neural network to jointly learn semantic and structural information of the code to detect statement-level memory-related vulnerability. Wang et al. [38] proposed a graph-based neural network DLVD model that focused on extracting class-separation features between vulnerable and non-vulnerable code. Wen et al. [39] proposed graph simplification and enhanced graph representation learning to effectively capture global information of code. Zhang et al. [48] utilized the CodeBERT and convolutional neural network to learn path representations from syntax-based control flow graph. However, in the face of class-imbalanced datasets, the effectiveness of DLVD methods will be reduced [5].

6.2 Empirical Research of Imbalanced Learning for Software Engineering Tasks

In the field of software engineering, there are many studies exploring the impact of data imbalance learning methods on software engineering tasks, such as data resampling [11, 12, 19, 41, 44, 53] , cost-sensitive learning [19, 45], and ensemble methods [44]. For instance, Zheng et al. [55] studied the class rebalancing methods to explore their impact on the performance of security error reporting prediction models. Li et al. [19] studied the impact of data resampling, cost-sensitive learning, and imbalanced ensemble learning on the data imbalance problem in code smell detection methods. Zhao et al. [53] studied resampling-based, ensemble-based, and cost-sensitive imbalanced learning techniques to understand their impact on model performance in crashing fault residence prediction task. Yang et al. [41] conducted an in-depth study of the impact of data resampling on DLVD when facing imbalanced datasets. However, undersampling methods may lead to the loss of valuable information, while oversampling methods may introduce overfitting problems [1, 27]. In addition, these methods may reduce the interpretability of the models [16, 27, 33–35]. Fu et al. [13] applied two imbalance loss functions (LALoss and FocalLoss) to train the vulnerability category classification models and found that these two loss functions can improve model performance. However, they only compared two imbalance loss functions for vulnerability category classification, and many other imbalance loss functions have not been investigated. Furthermore, no research has explored sufficient imbalance loss functions for vulnerability detection tasks and found which imbalance loss function is the best.

7 CONCLUSION

We present the first comprehensive study aimed at evaluating the impact of imbalance loss functions on two DLVD models to alleviate the class imbalance issue. Through a thorough examination of six widely used imbalance loss functions, CELoss, and two data resampling methods across three datasets, we analyze the performance of DLVD models in terms of Precision, Recall, and F1, employing statistical analysis with the Scott-Knott ESD test. Our experimental results indicate that LineVul consistently outperforms ReVeal across all datasets. Furthermore, LDAMLoss is recommended for achieving high Precision, while RUS is more suitable for obtaining high Recall. However, it is important to note that RUS may lead to information loss and reduced model interpretability. Consequently, we recommend LALoss, as it not only achieves high Recall but also delivers superior performance on the F1 metric. In conclusion, we advocate for the use of LineVul in conjunction with LALoss for DLVD.

Data Availability. Our source code and datasets are available at: *https://figshare.com/s/f6f3598a4d54887775e3*.

ACKNOWLEDGMENTS

This work is partially supported by the National Natural Science Foundation of China (No.62202350), the General Research Fund of the Research Grants Council of Hong Kong and the research funds of the City University of Hong Kong (No.6000796, No.9229109, No.9229098, No.9220103, and No.9229029), the Start-up Grant from

Wuhan University of Technology (No.104-40120693), and the National Training Program of Innovation and Entrepreneurship for Undergraduates (No.202410497045).

REFERENCES

[1] Kwabena Ebo Bennin, Jacky W Keung, and Akito Monden. 2019. On the relative value of data resampling approaches for software defect prediction. *Empirical Software Engineering* 24 (2019), 602–636.

[2] Tim Boland and Paul E Black. 2012. Juliet 1. 1 C/C++ and java test suite. *Computer* 45, 10 (2012), 88–90.

[3] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Aréchiga, and Tengyu Ma. 2019. Learning Imbalanced Datasets with Label-Distribution-Aware Margin Loss. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*. 1565–1576.

[4] Sicong Cao, Xiaobing Sun, Lili Bo, Rongxin Wu, Bin Li, and Chuanqi Tao. 2022. MVD: memory-related vulnerability detection based on flow-sensitive graph neural networks. In *44th IEEE International Conference on Software Engineering*. 1456–1468.

[5] Saikat Chakraborty, Rahul Krishna, Yangruibo Ding, and Baishakhi Ray. 2022. Deep learning based vulnerability detection: Are we there yet. *IEEE Transactions on Software Engineering* 48, 9 (2022), 3280–3296.

[6] Xiang Chen, Zhidan Yuan, Zhanqi Cui, Dun Zhang, and Xiaolin Ju. 2021. Empirical studies on the impact of filter-based ranking feature selection on security vulnerability prediction. *IET Software* 15, 1 (2021), 75–89.

[7] Xiang Chen, Yingquan Zhao, Zhanqi Cui, Guozhu Meng, Yang Liu, and Zan Wang. 2019. Large-scale empirical studies on effort-aware security vulnerability prediction methods. *IEEE Transactions on Reliability* 69, 1 (2019), 70–87.

[8] Yizhou Chen, Heng Dai, Xiao Yu, Wenhua Hu, Zhiwen Xie, and Cheng Tan. 2021. Improving Ponzi scheme contract detection using multi-channel TextCNN and transformer. *Sensors* 21, 19 (2021), 6417.

[9] Roland Croft, Muhammad Ali Babar, and M. Mehdi Kholoosi. 2023. Data Quality for Software Vulnerability Datasets. In *45th IEEE International Conference on Software Engineering*. IEEE, 121–133.

[10] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge J. Belongie. 2019. Class-Balanced Loss Based on Effective Number of Samples. In *IEEE Conference on Computer Vision and Pattern Recognition*. 9268–9277.

[11] Shuo Feng, Jacky Keung, Xiao Yu, Yan Xiao, Kwabena Ebo Bennin, Md Alamgir Kabir, and Miao Zhang. 2021. COSTE: Complexity-based OverSampling TEchnique to alleviate the class imbalance problem in software defect prediction. *Information and Software Technology* 129 (2021), 106432.

[12] Shuo Feng, Jacky Keung, Xiao Yu, Yan Xiao, and Miao Zhang. 2021. Investigation on the stability of SMOTE-based oversampling techniques in software defect prediction. *Information and Software Technology* 139 (2021), 106662.

[13] Michael Fu, Van Nguyen, Chakkrit Kla Tantithamthavorn, Trung Le, and Dinh Q. Phung. 2023. VulExplainer: A Transformer-Based Hierarchical Distillation for Explaining Vulnerability Types. *IEEE Transactions on Software Engineering* 49, 10 (2023), 4550–4565.

[14] Michael Fu and Chakkrit Tantithamthavorn. 2022. LineVul: A Transformer-based Line-Level Vulnerability Prediction. In *19th Mining Software Repositories*. 608–620.

[15] Shuzheng Gao, Cuiyun Gao, Yulan He, Jichuan Zeng, Lunyiu Nie, Xin Xia, and Michael Lyu. 2023. Code Structure–Guided Transformer for Source Code Summarization. *ACM Transactions on Software Engineering and Methodology* 32, 1 (2023), 1–32.

[16] Yuxiang Gao, Yi Zhu, and Yu Zhao. 2022. Dealing with imbalanced data for interpretable defect prediction. *Information and Software Technology* 151 (2022), 107016.

[17] Jirayus Jiarpakdee, Chakkrit Kla Tantithamthavorn, and John Grundy. 2021. Practitioners' perceptions of the goals and visual explanations of defect prediction models. In *18th International Conference on Mining Software Repositories*. 432–443.

[18] Buyu Li, Yu Liu, and Xiaogang Wang. 2019. Gradient harmonized single-stage detector. In *Proceedings of 33rd AAAI Conference on Artificial Intelligence*, Vol. 33. 8577–8584.

[19] Fuyang Li, Kuan Zou, Jacky Wai Keung, Xiao Yu, Shuo Feng, and Yan Xiao. 2023. On the relative value of imbalanced learning for code smell detection. *Software: Practice and Experience* 53, 10 (2023), 1902–1927.

[20] Yi Li, Shaohua Wang, and Tien N Nguyen. 2021. Vulnerability detection with fine-grained interpretations. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 292–303.

[21] Zhen Li, Deqing Zou, Shouhuai Xu, Hai Jin, Yawei Zhu, and Zhaoxuan Chen. 2021. Sysevr: A framework for using deep learning to detect software vulnerabilities. *IEEE Transactions on Dependable and Secure Computing* 19, 4 (2021), 2244–2258.

[22] Zhen Li, Deqing Zou, Shouhuai Xu, Xinyu Ou, Hai Jin, Sujuan Wang, Zhijun Deng, and Yuyi Zhong. 2018. VulDeePecker: A Deep Learning-Based System for Vulnerability Detection. In *25th Annual Network and Distributed System Security Symposium*.

[23] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. 2980–2988.

[24] Guilong Lu, Xiaolin Ju, Xiang Chen, Shaoyu Yang, Liang Chen, and Hao Shen. 2023. Assessing the Effectiveness of Vulnerability Detection via Prompt Tuning: An Empirical Study. In *30th Asia-Pacific Software Engineering Conference*. 415–424.

[25] Xiaoxue Ma, Jacky Keung, Zhen Yang, Xiao Yu, Yishu Li, and Hao Zhang. 2022. CASMS: Combining clustering with attention semantic model for identifying security bug reports. *Information and Software Technology* 147 (2022), 106906.

[26] Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. 2021. Long-tail learning via logit adjustment. In *9th International Conference on Learning Representations*.

[27] Giang Hoang Nguyen, Abdesselam Bouzerdoum, and Son Lam Phung. 2009. Learning pattern classification tasks with imbalanced data sets. *Pattern recognition* 10 (2009).

[28] Chao Ni, Xinrong Guo, Yan Zhu, Xiaodan Xu, and Xiaohu Yang. 2023. Function-Level Vulnerability Detection Through Fusing Multi-Modal Knowledge. In *38th International Conference on Automated Software Engineering*. 1911–1918.

[29] Chao Ni, Liyu Shen, Wei Wang, Xiang Chen, Xin Yin, and Lexiao Zhang. 2023. FVA: Assessing Function-Level Vulnerability by Integrating Flow-Sensitive Structure and Code Statement Semantic. In *31st IEEE International Conference on Program Comprehension*. 339–350.

[30] Yu Nong, Yuzhe Ou, Michael Pradel, Feng Chen, and Haipeng Cai. 2023. VULGEN: Realistic Vulnerability Generation Via Pattern Mining and Deep Learning. In *45th International Conference on Software Engineering*. 2527–2539.

[31] Reuven Rubinstein. 1999. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability* 1 (1999), 127–190.

[32] Rebecca Russell, Louis Kim, Lei Hamilton, Tomo Lazovich, Jacob Harer, Onur Ozdemir, Paul Ellingwood, and Marc McConley. 2018. Automated vulnerability detection in source code using deep representation learning. In *17th IEEE International Conference on Machine Learning and Applications*. 757–762.

[33] Chakkrit Tantithamthavorn and Ahmed E. Hassan. 2018. An experience report on defect modelling in practice: pitfalls and challenges. In *40th International Conference on Software Engineering: Software Engineering in Practice*. 286–295.

[34] Chakkrit Tantithamthavorn, Ahmed E Hassan, and Kenichi Matsumoto. 2018. The impact of class rebalancing techniques on the performance and interpretation of defect prediction models. *IEEE Transactions on Software Engineering* 46, 11 (2018), 1200–1219.

[35] Burak Turhan. 2012. On the dataset shift problem in software engineering prediction models. *Empirical Software Engineering* 17 (2012), 62–74.

[36] Ohad Volk and Gonen Singer. 2021. Adaptive cost-sensitive learning in neural networks for misclassification cost problems. *arXiv preprint arXiv:2111.07382* (2021).

[37] Sixuan Wang, Chen Huang, Dongjin Yu, and Xin Chen. 2023. VulGraB: Graph-embedding-based code vulnerability detection with bi-directional gated graph neural network. *Software: Practice and Experience* 53, 8 (2023), 1631–1658.

[38] Wenbo Wang, Tien N Nguyen, Shaohua Wang, Yi Li, Jiyuan Zhang, and Aashish Yadavally. 2023. DeepVD: Toward Class-Separation Features for Neural Network Vulnerability Detection. In *45th International Conference on Software Engineering*. 2249–2261.

[39] Xin-Cheng Wen, Yupan Chen, Cuiyun Gao, Hongyu Zhang, Jie M. Zhang, and Qing Liao. 2023. Vulnerability Detection with Graph Simplification and Enhanced Graph Representation Learning. In *45th International Conference on Software Engineering*. 2275–2286.

[40] Zhou Xu, Shuai Li, Jun Xu, Jin Liu, Xiapu Luo, Yifeng Zhang, Tao Zhang, Jacky Keung, and Yutian Tang. 2019. LDFR: Learning deep feature representation for software defect prediction. *Journal of Systems and Software* 158 (2019), 110402.

[41] Xu Yang, Shaowei Wang, Yi Li, and Shaohua Wang. 2023. Does data sampling improve deep learning-based vulnerability detection? Yeas! and Nays!. In *45th International Conference on Software Engineering*. 2287–2298.

[42] Zhen Yang, Jacky Keung, Xiao Yu, Xiaodong Gu, Zhengyuan Wei, Xiaoxue Ma, and Miao Zhang. 2021. A Multi-Modal Transformer-based Code Summarization Approach for Smart Contracts. In *29th International Conference on Program Comprehension*. IEEE, 1–12.

[43] Zhen Yang, Jacky Wai Keung, Xiao Yu, Yan Xiao, Zhi Jin, and Jingyu Zhang. 2023. On the significance of category prediction for code-comment synchronization. *ACM Transactions on Software Engineering and Methodology* 32, 2 (2023), 1–41.

[44] Xiao Yu, Jin Liu, Zijiang Yang, Xiangyang Jia, Qi Ling, and Sizhe Ye. 2017. Learning from imbalanced data for predicting the number of software defects. In *28th International Symposium on Software Reliability Engineering*. IEEE, 78–89.

[45] Zhe Yu, Christopher Theisen, Laurie Williams, and Tim Menzies. 2019. Improving vulnerability inspection efficiency using active learning. *IEEE Transactions on Software Engineering* 47, 11 (2019), 2401–2420.

[46] Chunyong Zhang, Bin Liu, Yang Xin, and Liangwei Yao. 2023. CPVD: Cross Project Vulnerability Detection Based On Graph Attention Network And Domain

Adaptation. *IEEE Transactions on Software Engineering* 49, 8 (2023), 4152–4168.

[47] Fengji Zhang, Xiao Yu, Jacky Keung, Fuyang Li, Zhiwen Xie, Zhen Yang, Caoyuan Ma, and Zhimin Zhang. 2022. Improving Stack Overflow question title generation with copying enhanced CodeBERT model and bi-modal information. *Information and Software Technology* 148 (2022), 106922.

[48] Junwei Zhang, Zhongxin Liu, Xing Hu, Xin Xia, and Shanping Li. 2023. Vulnerability Detection by Learning from Syntax-Based Execution Paths of Code. *IEEE Transactions on Software Engineering* 49, 8 (2023), 4196–4212.

[49] Xuejun Zhang, Fenghe Zhang, Bo Zhao, Bo Zhou, and Boyang Xiao. 2023. VulD-Transformer: Source Code Vulnerability Detection via Transformer. In *Proceedings of the 14th Asia-Pacific Symposium on Internetware*. 185–193.

[50] Yuting Zhang, Jiahao Zhu, Yixin Yang, Ming Wen, and Hai Jin. 2023. Comparing the Performance of Different Code Representations for Learning-based Vulnerability Detection. In *Proceedings of the 14th Asia-Pacific Symposium on Internetware*. 174–184.

[51] Zhuo Zhang, Yan Lei, Xiaoguang Mao, Meng Yan, Xin Xia, and David Lo. 2023. Context-Aware Neural Fault Localization. *IEEE Transactions on Software Engineering* 49, 7 (2023), 3939–3954.

[52] Zhuo Zhang, Yan Lei, Meng Yan, Yue Yu, Jiachi Chen, Shangwen Wang, and Xiaoguang Mao. 2022. Reentrancy vulnerability detection and localization: A deep learning based two-phase approach. In *37th International Conference on Automated Software Engineering*. 83:1–83:13.

[53] Kunsong Zhao, Zhou Xu, Meng Yan, Tao Zhang, Lei Xue, Ming Fan, and Jacky Keung. 2023. The impact of class imbalance techniques on crashing fault residence prediction models. *Empirical Software Engineering* 28, 2 (2023), 49.

[54] Wei Zheng, Jialiang Gao, Xiaoxue Wu, Fengyu Liu, Yuxing Xun, Guoliang Liu, and Xiang Chen. 2020. The impact factors on the performance of machine learning-based vulnerability detection: A comparative study. *Journal of Systems and Software* 168 (2020), 110659.

[55] Wei Zheng, Yuxing Xun, Xiaoxue Wu, Zhi Deng, Xiang Chen, and Yulei Sui. 2021. A comparative study of class rebalancing methods for security bug report classification. *IEEE Transactions on Reliability* 70, 4 (2021), 1658–1670.

[56] Yaqin Zhou, Shangqing Liu, Jing Kai Siow, Xiaoning Du, and Yang Liu. 2019. Devign: Effective Vulnerability Identification by Learning Comprehensive Program Semantics via Graph Neural Networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*. 10197–10207.