Tighter Clusters, Safer Code? Improving Vulnerability Detection with Enhanced Contrastive Loss

Pranav Kapparad[1] **Biju R Mohan**[1]
[1]National Institute of Technology Karnataka, Surathkal, India
pranavkapparad.211ai026@nitk.edu.in
biju@nitk.edu.in

Abstract

Distinguishing vulnerable code from non-vulnerable code is challenging due to high inter-class similarity. Supervised contrastive learning (SCL) improves embedding separation but struggles with intra-class clustering, especially when variations within the same class are subtle. We propose CLUSTER-ENHANCED SUPERVISED CONTRASTIVE LOSS (CESCL), an extension of SCL with a distance-based regularization term that tightens intra-class clustering while maintaining inter-class separation. Evaluating on CodeBERT and GraphCodeBERT with Binary Cross Entropy (BCE), BCE + SCL, and BCE + CESCL, our method improves F1 score by 1.76% on CodeBERT and 4.1% on GraphCodeBERT, demonstrating its effectiveness in code vulnerability detection and broader applicability to high-similarity classification tasks.

1 Introduction

Code vulnerability detection is a cornerstone of software security, particularly as the world undergoes rapid digitization. In domains like finance, healthcare, and government, software vulnerabilities exploited by malicious actors could have catastrophic consequences. The ability to detect such weaknesses efficiently is essential for safeguarding the trust that underpins modern technological systems. Beyond protecting sensitive data, robust vulnerability detection forms the backbone of a resilient digital society, ensuring confidence in the software solutions we rely on daily.

Recent years have witnessed a paradigm shift in this field with the rise of deep learning models, which have revolutionized how vulnerabilities are identified. These models far outperform traditional approaches such as static analysis tools and manual code reviews, which are labor-intensive, error-prone, and unable to keep pace with the accelerating rate of software development. Leveraging neural networks has enabled researchers to automate vulnerability detection, improving scalability and accuracy by identifying subtle patterns in code that signal potential flaws. Most deep learning models rely on loss functions such as binary cross-entropy to learn from labeled datasets of vulnerable and non-vulnerable code. However, despite these advancements, the field remains fraught with challenges.

One of the most significant bottlenecks in existing models is the high semantic and structural similarity between vulnerable and non-vulnerable code samples. This similarity often causes embeddings of the two classes to overlap in high-dimensional space, resulting in increased false positives and false negatives, thereby undermining the reliability of the models. To better quantify this challenge, we conducted a focused analysis of the embedding space. Using a simple, yet effective method, we computed the average cosine similarity between embeddings of samples with opposite labels across both code and general text datasets. The results, as seen in the figure below 1, revealed that code datasets exhibit significantly higher similarity across labels than general text data, underscoring the unique difficulty of separating vulnerable from non-vulnerable code. This inherent overlap in the embedding space presents a key challenge in ensuring accurate and reliable vulnerability detection.

Contrastive learning has emerged as a promising technique to address this challenge. By structuring the embedding space to maximize separation between samples with opposite labels and encouraging tighter clustering of samples within the same class, contrastive learning reduces overlap and enhances the discriminative power of embeddings. This is particularly critical in the context of code vulnerability detection, where subtle differences between classes demand a highly optimized embedding space.

However, existing contrastive learning methods,

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 4: Student Research Workshop), pages 247–252

such as Supervised Contrastive Loss (SCL), exhibit limitations in this domain. While SCL effectively prioritizes inter-class separation, it often fails to enforce sufficient intra-class cohesion. This can result in loosely clustered embeddings within each class, increasing the likelihood of misclassifications. Consequently, SCL struggles to handle the high similarity between vulnerable and non-vulnerable samples, limiting its effectiveness in real-world applications.

To overcome these limitations, we propose a novel loss function, Cluster Enhanced Supervised Contrastive Loss (CESCL). CESCL builds on the foundation of SCL by introducing additional regularization techniques aimed at simultaneously minimizing intra-class separation and penalizing high cosine similarity between embeddings of vulnerable and non-vulnerable code snippets. This dual objective ensures tighter clustering within the same class while amplifying the dissimilarity between different classes, resulting in a well-structured embedding space optimized for classification.

CESCL achieves this by incorporating penalties for misaligned embeddings and emphasizing the structural and semantic nuances that distinguish vulnerable from non-vulnerable code. By fostering tighter intra-class cohesion and greater inter-class separation, CESCL reduces embedding overlap, enabling models to better generalize across diverse and unseen code patterns. This results in lower false positive and false negative rates, addressing key reliability concerns in existing systems.

In summary, this research introduces CESCL as a targeted solution to the embedding challenges in code vulnerability detection. By addressing the shortcomings of existing loss functions, CESCL provides a more robust and generalizable embedding space, significantly improving classification accuracy. Our work also highlights the unique challenges of this domain through a quantitative analysis of embedding similarity, offering a new perspective on the limitations of current approaches.

As software vulnerabilities continue to rise alongside the pace of digitization, the need for reliable and efficient detection methods has become more urgent than ever. CESCL represents a step forward in building secure and trustworthy software systems, offering a foundation for future advancements in vulnerability detection. By bridging the gap between the limitations of SCL and the demands of real-world applications, this research provides both a theoretical and practical contribution to the field, paving the way for more secure digital ecosystems.

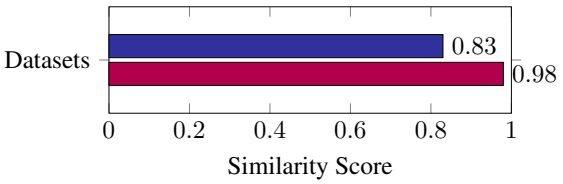

Figure 1: Cosine Similarity Between Opposite Labels. Text Data (Positive vs. Negative) is represented in blue, while Code Data (Vulnerable vs. Non-Vulnerable) is represented in red.

2 Related Work

Code vulnerability detection using deep learning has gained quite some attention in recent years, with various methods developed to address the challenges posed by detecting vulnerabilities within code (Grieco et al. (2016); Lin et al. (2017)). Early approaches Li et al. (2018), make use Long Bi Short-Term Memory (Bi-LSTM) networks to analyze code based on sequences of API calls, exhibiting the efficacy of deep learning models in capturing common patterns associated with vulnerbale code. SySeVR Li et al. (2021), built upon this approach by developing a deep learning framework for detecting vulnerabilities through sequence modeling of vulnerable function calls. While these methods provide useful insights, they often struggle with capturing the broader structural and semantic complexities of code, restricting their performance on more sophisticated code samples.

More recently, transformer-based models, which make use of attention mechanism, like CodeBERT Feng et al. (2020) and GraphCodeBERT Guo et al. (2020) have brought about major advancements. CodeBert is a pretrained model tailor made for both programming and natural languages, capturing both syntactic and semantic features from a large corpus of code. It has been adopted widely for vulnerability detection because of its ability to handle tasks like code summarization, generation, and classification. GraphCodeBERT extends CodeBert by incorporating data flow information within the model's architecture. This approach enhances the model's understanding of dependencies and control structures, allowing it to detect vulnerabilities that rely on intricate code flows, an area where traditional transformer models tend to disappoint. Such advancements highlight the potential

of transformer based models in advancing the field of code vulnerability detection.

The Devign dataset Zhou et al. (2019) has been crucial in evaluating and benchmarking vulnerability detection models. Devign contains over 21,000 labeled C/C++ code snippets drawn from open-source projects, with each snippet classified as vulnerable or non-vulnerable. The dataset presents unique challenges due to the incorporation of a wide variety of vulnerabilities. The dataset also constitutes complex vulnerabilities making it hard for models to effectively generalise across samples. Devign also provides a rich structural context, including abstract syntax trees (ASTs) and control flow graphs (CFGs), which has proven useful for models designed to capture graph-based relationships in code, as demonstrated in Zhou et al. (2019) original work making use of graph neural networks.

Supervised Contrastive Learning (SCL), introduced by Khosla et al. (2020), is an emerging and powerful technique that focuses class separation by leveraging both positive and negative samples, making it immensely suitable for tasks where closely related samples are to be differentiated. In vulnerability detection, where samples of vulnerable and non vulnerable can appear notoriously similar, SCL has shown potential Du et al. (2022) by promoting embedding spaces that separate vulnerable and non-vulnerable samples.

Various regularization techniques have been applied to these losses to improve robustness in high-similarity domains such as the one tackled in this paper. For instance, Botev et al. (2022) explored regularizing for invariance to data augmentation, improving the ability of models to handle difficult samples.

In summary, this study builds on the strengths of transformer models like CodeBert and GraphCodeBERT, evaluates performance on the Devign dataset, and explores advanced contrastive learning techniques to enhance code vulnerability detection. By combining supervised contrastive learning with regularization strategies, we aim to improve the model's capability in embedding separation, thus enhancing overall classification performance.

3 Methodology

To ameliorate the effectiveness of code vulnerability detection, this study builds a classification model on top of the both CodeBERT and GraphCodeBERT models, with additional dropout and batch normalization layers. These layers reduce overfitting and ensure stable training by normalizing activations, contributing to more reliable model performance. The central novelty in this approach is the novel loss function, which integrates supervised contrastive learning with a distance-based regularization term to improve embedding separation between classes.

3.1 Dataset

For this research work, we make use of the Devign dataset, a benchmark dataset for code vulnerability detection in C/C++ programs. The Devign dataset constitutes over 21,000 code snippets, each labeled as either vulnerable or non-vulnerable, collected from real-world open-source projects, FFmpeg and qemu. Each code snippet is annotated with several features, including abstract syntax tree (AST) representations, control flow graphs (CFGs), and data flow information, which capture both structural and semantic information essential for identifying vulnerabilities. For this work, only the code function is made use of since the focus is on the embedding separation.

3.2 Model Architecture

The architecture begins with a pre-trained model, either CodeBERT or GraphCodeBERT, which is fine-tuned on domain-specific code datasets to generate meaningful code embeddings. On top of these embeddings, a classifier head consisting of fully connected layers is added. The classifier head takes as input a tensor of size 768 (the embedding output of CodeBERT or GraphCodeBERT), followed by a 128-dimensional layer, and finally an output layer of size 1. Dropout layers are interleaved within the classifier to reduce overfitting by randomly deactivating neurons during training, and batch normalization layers are employed to stabilize and accelerate the training process by standardizing layer inputs. The model output provides a binary classification, predicting whether a code snippet is vulnerable.

3.3 Loss Function Design

The novel contribution of this work is a custom loss function, cluster enhanced supervised contrastive loss (CESCL), that enhances embedding separation. This function combines the supervised contrastive loss (SCL loss) with a distance-based regularization term, which encourages tighter clustering within each class. The components of this loss

function are as follows:

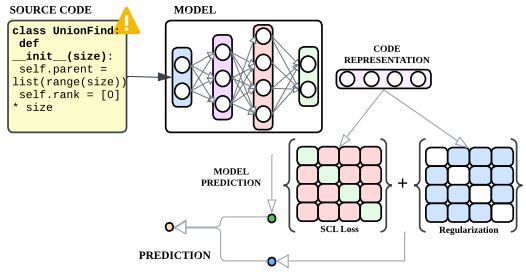

Figure 2: The framework of the proposed method.

Supervised Contrastive Loss (SCL): This loss maximizes agreement and similarity between embeddings of samples within the same class while pushing apart embeddings of different classes apart. Specifically, feature vectors are normalized, and a contrastive logits matrix is computed by dividing the dot product of normalized feature vectors by a temperature scaling factor.The novel contribution of this work is a custom loss function, cluster enhanced supervised contrastive loss (CESCL), that enhances embedding separation. This function combines the supervised contrastive loss (SCL loss) with a distance-based regularization term, which encourages tighter clustering within each class. The formula (Khosla et al., 2020) is as below

$$\mathcal{L}_{\text{SCL}} = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{|P(i)|}\sum_{p\in P(i)}\log\frac{\exp\left(\frac{\mathbf{z}_i\cdot\mathbf{z}_p}{\tau}\right)}{\sum_{a\in A(i)}\exp\left(\frac{\mathbf{z}_i\cdot\mathbf{z}_a}{\tau}\right)} \tag{1}$$

where:

- N is the batch size,
- $P(i)$ represents the set of positive samples for anchor i,
- $A(i)$ represents the set of all samples in the batch excluding i,
- $\mathbf{z}_i$ and $\mathbf{z}_p$ are the normalized feature vectors of the anchor and positive samples, respectively,
- τ is the temperature scaling factor, which helps control the distribution of the similarity scores.

While SCL effectively separates different classes, it does not explicitly enforce compactness within the same class, leading to loosely clustered embeddings. This is particularly problematic in high-similarity domains like vulnerability detection, where even minor variations can mislead classification.

Distance-Based Regularization Term: To further improve intra-class clustering, a regularization term is added that penalizes large distances between embeddings within the same class. This regularization term calculates the pairwise Euclidean distances between embeddings of the same class, averaging them over all possible pairs, and is scaled by a regularization factor.

The formula for the regularisation is as below

$$\mathcal{L}_{\text{reg}} = \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}1_{[L(i)=L(j)]}\|\mathbf{z}_i-\mathbf{z}_j\|^2 \tag{2}$$

where:

- n is the total number of samples,
- $1_{[L(i)=L(j)]}$ is an indicator function, equal to 1 if samples i and j belong to the same class, and 0 otherwise,
- $\mathbf{z}_i$ and $\mathbf{z}_j$ are the feature vectors of samples i and j.

Cluster Enhanced Supervised Contrastive Loss is a combination of Supervised Contrastive Loss and the Distance Based Regularization Term (2). It is as below

$$\mathcal{L}_{\text{Cluster-Enhanced SCL}} = \mathcal{L}_{\text{SCL}} + \lambda_{\text{reg}}\cdot\mathcal{L}_{\text{reg}} \tag{3}$$

where λ_{reg} is a hyperparameter that scales the contribution of the regularization term.

3.4 Training

The model is trained using the combined loss function, which integrates the Binary Cross Entropy (BCE) loss with the Cluster-Enhanced Supervised Contrastive Loss. Specifically, the final loss is computed as:

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{BCE}} + \alpha\cdot\mathcal{L}_{\text{Cluster-Enhanced SCL}}, \tag{4}$$

where α is a balancing hyperparameter. The Cluster-Enhanced Supervised Contrastive Loss is defined as:

$$\mathcal{L}_{\text{Cluster-Enhanced SCL}} = \mathcal{L}_{\text{SCL}} + \lambda_{\text{reg}}\cdot\mathcal{L}_{\text{reg}} \tag{5}$$

where λ_{reg} is a hyperparameter that scales the contribution of the regularization term.

In our experiments, we set $\lambda_{\text{reg}} = 0.5$ and $\alpha = 0.2$ based on preliminary grid search evaluations.

4 Results and Analysis

In this study, three models were trained to assess the impact of different loss functions on code vulnerability detection. Each model uses the same architecture, a classifier built on top of a pre-trained CodeBERT or GraphCodeBERT model with dropout and batch normalization. The only difference between models being the loss function utilized during training:

- **Model 1:** Binary Cross Entropy (BCE) loss only.
- **Model 2:** BCE combined with Supervised Contrastive Loss (SCL).
- **Model 3:** BCE combined with Cluster Enhanced Supervised Contrastive Loss (CESCL).

To evaluate performance, F1 scores were calculated on the test set for each model. These scores provide a comparison between each model's precision and recall, informing how well the different loss functions contribute to the model's accuracy and embedding separation.

On top of this, the silhouette score was calculated as a measure of embedding separation. The silhouette score is a widely-used metric to evaluate clustering quality. It ranges from -1 to 1, where a value near 1 indicates that samples are well-separated and closely grouped within their respective clusters, and a value near -1 suggests significant overlap between clusters. In the context of our study, a higher silhouette score implies that code snippets belonging to the same class (vulnerable or non-vulnerable) are more similar to each other than to those in the opposing class, thereby indicating effective embedding separation.

Table 1: Performance Comparison of Models (F1 Score)

Model	F1 Score
CodeBERT	0.597
CodeBERT + SCL	0.614
CodeBERT + CESCL	0.625
GraphCodeBERT	0.594
GraphCodeBERT + SCL	0.607
GraphCodeBERT + CESCL	0.633

Table 2: Performance Comparison of Models (Silhouette Score)

Model	Silhouette Score
CodeBERT	0.052
CodeBERT + SCL	0.043
CodeBERT + CESCL	0.056
GraphCodeBERT	0.046
GraphCodeBERT + SCL	0.031
GraphCodeBERT + CESCL	0.050

5 Conclusion

In this work, we introduced Cluster-Enhanced Supervised Contrastive Loss (CESCL), a novel loss function designed to improve the embedding quality for code vulnerability detection. We evaluated the performance of CESCL in combination with both CodeBERT and GraphCodeBERT architectures, comparing it with the standard Binary Cross-Entropy (BCE) and BCE + Supervised Contrastive Learning (SCL) models.

The experimental results, as shown in Tables 1 and 2, demonstrate that CESCL consistently outperforms both BCE and BCE + SCL across the models tested. Notably, CodeBERT + CESCL achieved the highest F1 score of 0.625 among CodeBERT models and the best Silhouette score of 0.056, highlighting its ability to generate well-clustered and discriminative embeddings for vulnerability classification. Similarly, GraphCodeBERT + CESCL showed significant improvements, achieving a 4.1% increase in F1 score and a favorable Silhouette score of 0.050 compared to the other configurations.

It is worth noting that, although the incorporation of SCL in isolation sometimes led to a reduction in the silhouette score, the overall improvement in F1 score indicates that the CESCL framework effectively optimizes the embedding space for classification. This discrepancy suggests that while the silhouette score is a useful measure of clustering quality, it may not fully capture the nuances that contribute to enhanced detection performance in this context.

These results indicate that CESCL effectively improves intra-class compactness and inter-class separation, thereby enhancing the performance of the model in detecting vulnerabilities in code. Future work could explore further optimizations to the CESCL framework and test it on additional code-related tasks to fully realize its potential in improving the robustness and reliability of code classification models.

References

Aleksander Botev, Matthias Bauer, and Soham De. 2022. Regularising for invariance to data augmentation improves supervised learning. *arXiv preprint arXiv:2203.03304*.

Qianjin Du, Xiaohui Kuang, and Gang Zhao. 2022. Code vulnerability detection via nearest neighbor mechanism. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6173–6178.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. 2020. Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*.

Gustavo Grieco, Guillermo Luis Grinblat, Lucas Uzal, Sanjay Rawat, Josselin Feist, and Laurent Mounier. 2016. Toward large-scale vulnerability discovery using machine learning. In *Proceedings of the sixth ACM conference on data and application security and privacy*, pages 85–96.

Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, Shujie Liu, Long Zhou, Nan Duan, Alexey Svyatkovskiy, Shengyu Fu, et al. 2020. Graphcodebert: Pre-training code representations with data flow. *arXiv preprint arXiv:2009.08366*.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673.

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):2244–2258.

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 *Proceedings 2018 Network and Distributed System Security Symposium*, NDSS 2018. Internet Society.

Guanjun Lin, Jun Zhang, Wei Luo, Lei Pan, and Yang Xiang. 2017. Poster: Vulnerability discovery with function representation learning from unlabeled projects. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 2539–2541.

Yaqin Zhou, Shangqing Liu, Jingkai Siow, Xiaoning Du, and Yang Liu. 2019. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. *Advances in neural information processing systems*, 32.