# Vulnerability Detection via Topological Analysis of Attention Maps

Snopov P.[1], Golubinsky A.N.[1]

Institute for Information Transmission Problems of Russian Academy of Sciences
snopov@iitp.ru

**Abstract.** Recently, deep learning (DL) approaches to vulnerability detection have gained significant traction. These methods demonstrate promising results, often surpassing traditional static code analysis tools in effectiveness.

In this study, we explore a novel approach to vulnerability detection utilizing the tools from topological data analysis (TDA) on the attention matrices of the BERT model. Our findings reveal that traditional machine learning (ML) techniques, when trained on the topological features extracted from these attention matrices, can perform competitively with pre-trained language models (LLMs) such as CodeBERTa. This suggests that TDA tools, including persistent homology, are capable of effectively capturing semantic information critical for identifying vulnerabilities.

**Keywords:** Vulnerability detection, Persistent homology, Large language models

## 1 Introduction

The problem of source code vulnerability detection has become increasingly important in the field of software engineering. This complex task involves analyzing both the syntax and semantics of the source code. Traditionally, static analysis methods have dominated the field of vulnerability detection. These static tools tend to rely heavily on the syntax of the program, thereby overlooking potential information hidden in the semantics. As a result, such approaches often suffer from high rates of false positives.

Recently, however, various machine learning (ML) and deep learning (DL) methods have emerged [14,?]. Most DL-based solutions leverage graph neural networks (GNNs) and large language models (LLMs). These methods aim to learn both the semantics and syntax of the program. They have shown promising results, outperforming traditional static analysis tools and demonstrating low rates of false positives and false negatives [7].

Our work presents a novel approach to the vulnerability detection problem. As previously mentioned, the superiority of DL-based models over static ones is potentially due to their ability to capture the semantic information of the program. This semantic information is captured by the neural networks during training or fine-tuning. Furthermore, LLM-based models can capture this information even without fine-tuning.

On the other hand, the semantics of the source code, like its syntax, can also be represented as a tree or a graph. This representation allows for the application of topological methods.

In this work, we leverage the BERT model, pre-trained on source code, to capture the semantic information of programs. We also utilize tools from topological data analysis (TDA) [4] to explore the relationships within this semantic information. Our aim is to analyze and interpret the attention matrices using topological instruments.

Our work is inspired by the work of Laida Kushareva et. al. [9], in which the authors apply the topological methods for the artificial text detection. They demonstrated that the topology of the semantics captured by an LLM model (such as BERT) provides sufficient information for successful classification. Their approach outperformed neural baselines and performed on par with a fully fine-tuned BERT model, while being more robust to unseen data.

## 2 Background

### 2.1 BERT Model

BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based language model that has set new benchmarks in various natural language processing (NLP) tasks. The BERT architecture consists of $L$ encoder layers, each with $H$ attention heads. Each attention head receives a matrix $X$ as input, representing the $d$-dimensional embeddings of $m$ tokens, so that $X$ is of shape $m \times d$. The head outputs an updated representation matrix $X^{\text{out}}$:

$$X^{\text{out}} = W^{\text{attn}}(XW^V), \text{ with } W^{\text{attn}} = \text{softmax}\Big(\frac{(XW^Q)(XW^K)^T}{\sqrt{d}}\Big). \quad (1)$$

Here $W^Q, W^K, W^V$ are trained projection matrices of shape $d \times d$ and $W^{\text{attn}}$ is matrix of attention weights with shape $m \times m$ Each element $w_{ij}^{\text{attn}}$ can be interpreted as a weight of the $j$-th input's *relation* to the $i$-th output where larger weights indicate a stronger connection between the two tokens.

### 2.2 Attention Graph

An *attention graph* is a weighted graph representation of an attention matrix $W^{attn}$, where vertices represent the tokens and the edges connect a pair of tokens if the corresponding weight exceeds a predefined threshold value.

Setting this threshold value is critical yet challenging, as it distinguishes weak and strong relations between tokens. Additionally, varying the threshold can significantly alter the graph structure.

Topological data analysis (TDA) methods can extract properties of the graph structure without specifying an exact threshold value, addressing this challenge effectively.

### 2.3 Topological Data Analysis

Topological data analysis (TDA) is an emerging field that applies algebraic topology methods to data science. There are numerous excellent tutorials and surveys available for both non-mathematicians [10,?] and those with a mathematical background [5,?,?].

The main tool in topological data analysis, *persistent homology*, tracks changes in the topological structure of various objects, such as point clouds, scalar functions, images, and weighted graphs [1,?].

In our work, given a set of tokens $V$ and an attention matrix $W$,we construct a family of attention graphs indexed by increasing threshold values. This family, known as a *filtration*, is a fundamental object in TDA

With this sequence of graphs, we compute persistent homology in dimensions 0 and 1. Dimension 0 reveals connected components or clusters in the data, while dimension 1 identifies cycles or «loops». These computations yield a *persistence diagram*, which can be used to derive specific topological features, such as the number of connected components and cycles (such features are also called the *Betti numbers*) (see Appendix B for the details).

## 3 Topological Features of the Attention Graphs

For each code sample, we calculate the persistent homology in dimensions 0 and 1 of the symmetrized attention matrices, obtaining the persistence diagram for each attention head of the BERT model. We compute the following features in each dimension from the diagrams:

- Mean lifespan of points on the diagram
- Variance of the lifespan of points on the diagram
- Maximum lifespan of points on the diagram
- Total number of points on the diagram
- Persistence entropy

We symmetrize attention matrices to enable the application of persistent homology techniques. Symmetrizing attention matrices allows us to interpret them as distance matrices of a point cloud embedded in Euclidean space. We symmetrize attention matrices as follows:

$$\forall i, j : W_{ij}^{\text{sym}} = \max\left(W_{ij}^{\text{attn}}, W_{ji}^{\text{attn}}\right). \tag{2}$$

Alternatively, one can think of attention graphs, in which an edge between the vertices $i$ and $j$ appears if the threshold is greater than both $W_{ij}^{\text{attn}}$ and $W_{ji}^{\text{attn}}$.

We consider these features asthe numerical characteristics of the semantic evolution processes in the attention heads. These features encode the information about the clusters of mutual influence of the tokens in the sentence and the local structures like cycles. The features with «significant» persistence (i.e. those with large lifespan) correspond to the stable processes, whileas the features with short lifespans are highly susceptible to noise and do not reflect the stable topological attributes.

## 4 Experiments

**Methodology** To evaluate whether the encoded topological information can be used for vulnerability detection, we train Logistic Regression, Support Vector Machine (SVM), and Gradient Boosting classifiers on the topological features derived from the attention matrices of the BERT model, as described in Section 3. We utilize the *scikit-learn* library [12] for Logistic Regression and SVM, and the *LightGBM* library [8] for Gradient Boosting. Detailed training procedures are outlined in Appendix A.

**Data** We train and evaluate our classifier on *Devign* dataset. This dataset comprises samples from two large, widely-used open-source C-language projects: QEMU, and FFmpeg, which are popular among developers and diverse in functionality. Due to computational constraints, we were only using those data samples, that, being tokenized, are of length less than 150. This ensures that the point cloud constructed during attention symmetrization is also limited to a maximum length of 150.

**Baselines** We employ the `microsoft/codebert-base` model [6] from the HuggingFace library [16] as our pre-trained BERT-based baseline. Additionally, we fully fine-tune the `microsoft/codebert-base` model for comparison.

## 5 Results and Discussion

Table 1 outlines the results of the vulnerability detection experiments on the *Devign* dataset. The results reveal that the proposed topology-based classifiers outperform the chosen large language model (LLM) without fine-tuning but perform worse than the fine-tuned version.

**Table 1.** The results of the vulnerability detection experiments.

| **Model** | **F1 score** | **Accuracy** |
|---|---|---|
| Logistic Regression | 0.22 | 0.54 |
| LightGBM | **0.55** | 0.63 |
| SVM | 0.54 | **0.65** |
| CodeBERTa (pre-trained) | 0.28 | 0.45 |
| CodeBERTa (fine-tuned) | **0.71** | **0.72** |

These observations indicate that the information about a code snippet's vulnerability is encoded in the topological attributes of the attention matrices. The semantic evolution in the attention heads reflects code properties that are crucial for the vulnerability detection task, and persistent homology proves to be an effective method for extracting this information.

Notably, only the semantic information from attention heads was used. The inclusion of additional topological features obtained from the structural information of the source code, such as the topology of graph representations of the source code, could potentially enhance the overall performance of the proposed models.

## 6 Consclusion

This paper introduces a novel approach for the vulnerability detection task based on topological data analysis (TDA). We propose a set of interpretable topological features, obtained from persistence diagrams derived from the attention matrices of any transformer-based language model. The experiments demonstrate that machine learning classifiers trained on these features perform comparably to pre-trained code-specific large language models (LLMs).

We are making our code publicly available[1] with the aim of encouraging research into TDA-based methods for vulnerability detection and other NLP tasks. A potential direction for future work is to combine topological features that encode semantics with topological features that encode structural information. It is also interesting to investigate the topology of different symmetrizations of the attention matrices and how they encode semantics. Another promising direction is to incorporate multiparameter persistent homology to study the semantic evolution in the attention heads.

**Acknowledgements** The work was carried out as part of the state assignment N 1021061609772-0-1.2.1 (FFNU-2024-0020).

## References

1. Adams, H., Chepushtanova, S., Emerson, T., Hanson, E., Kirby, M., Motta, F., Neville, R., Peterson, C., Shipman, P., Ziegelmeier, L.: Persistence images: A stable vector representation of persistent homology (2016), https://arxiv.org/abs/1507.06217
2. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework (2019), https://arxiv.org/abs/1907.10902
3. Aktas, M.E., Akbas, E., Fatmaoui, A.E.: Persistence homology of networks: methods and applications. Applied Network Science **4**(1) (Aug 2019). https://doi.org/10.1007/s41109-019-0179-3, http://dx.doi.org/10.1007/s41109-019-0179-3
4. Chazal, F., Michel, B.: An introduction to topological data analysis: Fundamental and practical aspects for data scientists. Frontiers in Artificial Intelligence **4** (2021). https://doi.org/10.3389/frai.2021.667963, https://www.frontiersin.org/articles/10.3389/frai.2021.667963
5. Edelsbrunner, H., Harer, J.: Persistent homology—a survey (2008). https://doi.org/10.1090/conm/453/08802, http://dx.doi.org/10.1090/conm/453/08802

[1] link

6. Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., Zhou, M.: Codebert: A pre-trained model for programming and natural languages (2020)
7. Katsadouros, E., Patrikakis, C.: A Survey on Vulnerability Prediction using GNNs. In: Proceedings of the 26th Pan-Hellenic Conference on Informatics. pp. 38–43. ACM, Athens Greece (nov 2022). https://doi.org/10.1145/3575879.3575964, https://dl.acm.org/doi/10.1145/3575879.3575964
8. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems **30**, 3146–3154 (2017)
9. Kushnareva, L., Cherniavskii, D., Mikhailov, V., Artemova, E., Barannikov, S., Bernstein, A., Piontkovskaya, I., Piontkovski, D., Burnaev, E.: Artificial Text Detection via Examining the Topology of Attention Maps. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. pp. 635–649 (2021). https://doi.org/10.18653/v1/2021.emnlp-main.50, http://arxiv.org/abs/2109.04825, arXiv:2109.04825 [cs, math]
10. Murugan, J., Robertson, D.: An introduction to topological data analysis for physicists: From lgm to frbs (2019), https://arxiv.org/abs/1904.11044
11. Oudot, S.: Persistence Theory: From Quiver Representations to Data Analysis. American Mathematical Society (Dec 2015). https://doi.org/10.1090/surv/209, http://dx.doi.org/10.1090/surv/209
12. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
13. Schenck, H.: Algebraic Foundations for Applied Topology and Data Analysis. Springer International Publishing (2022). https://doi.org/10.1007/978-3-031-06664-1, http://dx.doi.org/10.1007/978-3-031-06664-1
14. Sharma, T., Kechagia, M., Georgiou, S., Tiwari, R., Vats, I., Moazen, H., Sarro, F.: A survey on machine learning techniques applied to source code. Journal of Systems and Software **209**, 111934 (mar 2024). https://doi.org/10.1016/j.jss.2023.111934, https://www.sciencedirect.com/science/article/pii/S0164121223003291
15. Steenhoek, B., Rahman, M.M., Jiles, R., Le, W.: An Empirical Study of Deep Learning Models for Vulnerability Detection (feb 2023). https://doi.org/10.48550/arXiv.2212.08109, http://arxiv.org/abs/2212.08109, arXiv:2212.08109 [cs]
16. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Huggingface's transformers: State-of-the-art natural language processing (2020), https://arxiv.org/abs/1910.03771

# Appendix A Training Details

**Fine-tuning CodeBERTa** We trained with the cosine scheduler with an initial learning rate $\mathrm{lr} = 5e - 5$ and set the number of epochs $\mathrm{e} = 15$.

**Hyperparameter search** We employed *Optuna* [2] to find optimal hyperparameters for both SVM and LightGBM models.

For SVM, the optimal hyperparameters were $C = 9.97$, $\gamma = \mathrm{auto}$ and $\mathrm{kernel} = \mathrm{rbf}$.

For LightGBM, the optimal parameters were $\lambda_{\ell_1} = 5.97$, $\lambda_{\ell_2} = 0.05$, $\mathrm{num\_leaves} = 422$, $\mathrm{feature\_fraction} = 0.65$, $\mathrm{bagging\_fraction} = 0.93$, $\mathrm{bagging\_freq} = 15$, $\mathrm{min\_child\_samples} = 21$.

# Appendix B Persistent Homology

Recall that a simplicial complex $X$ is a collection of $p$-dimensional simplices, i.e., vertices, edges, triangles, tetrahedrons, and so on. Simplicial complexes generalize graphs, which consist of vertices (0-simplices) and edges (1-simplices) and can represent higher-order interactions. A family of increasing simplicial complexes

$$\emptyset \subseteq X_0 \subseteq X_1 \subseteq \cdots \subseteq X_{n-1} \subseteq X_n$$

is called a *filtration*.

The idea of *persistence* involves tracking the evolution of simplicial complexes over the filtration. *Persistent homology* allows to trace the changes in homology vector spaces[2] of simplicial complexes that are present in the filtration. Given a filtration $\{X_t\}_{t=0}^n$, the homology functor $H_p$ applied to the filtration generates a sequence of vector spaces $H_p(X_t)$ and maps $i_*$ between them

$$H_p(X_*) : H_p(X_0) \xrightarrow{i_0} H_p(X_1) \xrightarrow{i_1} \cdots \xrightarrow{i_{n-1}} H_p(X_n).$$

Each vector spaces encodes information about the simplicial complex $X$ and its subcomplexes $X_i$. For example, $H_0$ generally encodes the connectivity of the space (or, in data terms, $H_0$ encodes the clusters of data), $H_1$ encodes the presence of 1-cycles, i.e., loops, $H_2$ represents the presence of 2-cycles, and so on. Persistence tracks the generators of each vector space through the induced maps. Some generators will vanish, while others will persist. Those that persist are likely the most important, as they represent features that truly exist in $X$. Therefore, persistent homology allows one to gain information about the underlying topological space via the sequence of its subspaces, the filtration.

[2] Usually, homology groups are considered with integral coefficients, but in the realm of persistence, homology groups are taken with coefficients in some field, for example, $\mathbb{Z}_p$ for some large prime number $p$. Hence, the homology groups become homology vector spaces.

In algebraic terms, the sequence of vector spaces $H_p(X_t)$ and maps $i_*$ between them can be seen as a representation of a quiver $I_n$. From the representation theory of quivers it is known, due to Gabriel, that any such representation is isomorphic to a direct sum of indecomposable interval representations $I[b_i, d_i]$, that is,

$$H_p(X_*) \simeq \bigoplus_i I[b_i, d_i].$$

The pairs $(b_i, d_i)$ represent the persistence of topological features, where $b_i$ denotes the time of birth and $d_i$ denotes the time of death of the feature. These pairs can be visualized via *barcodes* where each bar starts at $b_i$ and ends at $d_i$.

This information is also commonly represented using *persistence diagrams.* A persistence diagram is a (multi)set of points in the extended plane $\overline{\mathbb{R}^2}$, which reflects the structure of persistent homology. Given a set of pairs $(b_i, d_i)$, each pair can be considered as a point in the diagram with coordinates $(b_i, d_i)$. Thus, a persistence diagram is defined as

$$\mathrm{dgm}(H_p(X_*)) = \{(b_i, d_i) : I[b_i, d_i] \text{ is a direct summand in } H_p(X_*)\}.$$

Persistence diagrams provide a detailed visual representation of the topology of point clouds. However, integrating them into machine learning models presents significant challenges due to their complex structure. To effectively use the information from persistence diagrams in predictive models, it is crucial to transform the data into a suitable format, such as by applying persistence entropy.

*Persistence entropy* is a specialized form of Shannon entropy specially designed for persistence diagrams and is calculated as follows:

$$PE_k(X) \coloneqq - \sum_{(b_i, d_i) \in D_k} p_i \log(p_i),$$

where

$$p_i = \frac{d_i - b_i}{\sum_{(b_i, d_i) \in D_k} (d_i - b_i)} \quad \text{and} \quad D_k \coloneqq \mathrm{dgm}(H_k(X_\bullet)).$$

This numerical characteristic of a persistence diagram has several advantageous properties. Notably, it is stable under certain mild assumptions. This stability means there is a bound that «controls» the perturbations caused by noise in the input data.