Beyond Probing: Causal Analysis of Layer-wise Representations in Code Language Models

Anonymous ACL submission

Abstract

Pre-trained code language models achieve strong performance on code understanding tasks, yet it remains unclear how representations across layers contribute to model predictions. Prior work largely relies on linear probing, which measures whether task-relevant information is present in hidden states but does not indicate whether it is actually used during inference.

In this paper, we analyze layer-wise representations in both encoder-only and decoder-only models on vulnerability detection and code retrieval tasks. In addition to probing, we apply activation-based interventions, including patching and masking, to examine how intermediate representations affect model predictions. We observe a mismatch between what probing captures and what influences model outputs: layers with strong probing performance do not always lead to large prediction changes under intervention. For vulnerability detection, we observe an asymmetry, where removing vulnerability-related features shifts predictions, while adding them has a smaller effect. This suggests that vulnerability signals are more directly tied to model outputs. For code retrieval, we observe that strong intermediate representations do not necessarily translate into effective retrieval performance without task-specific alignment. This result further supports that information encoded in hidden states may remain inaccessible or functionally unused under the original model objective. Overall, these results indicate that probing alone does not fully capture how representations are used and that intervention-based analysis provides a complementary view.

1 Introduction

Transformer-based pre-trained models (Vaswani et al., 2017; Devlin et al., 2019) have achieved strong performance on natural language and code understanding tasks. In software engineering, models such as CodeBERT (Feng et al., 2020) and CodeT5 (Wang et al., 2021) perform well on vulnerability detection and code retrieval benchmarks. Despite these results, it remains unclear how these models organize and use task-relevant information internally. This is particularly important in security settings, where understanding why a model predicts a vulnerability matters as much as the prediction itself.

Linear probing (Alain and Bengio, 2018; Adi et al., 2017) is widely used to study internal representations by testing whether information can be extracted from hidden states. Prior work shows that code models encode useful semantic features at intermediate layers. However, probing only establishes the presence of information, not whether the model relies on it during prediction (Hewitt and Manning, 2019; Elazar et al., 2021; Belinkov, 2022). As a result, strong probing performance does not necessarily reflect the mechanisms driving model behavior.

To study how intermediate representations affect predictions, we perform a causal analysis using activation patching (Meng et al., 2022). We focus on encoder-only and decoder-only models for vulnerability detection, using paired vulnerable and fixed code samples. During inference, we perturb representations at the locations corresponding to vulnerability fixes, and measure the resulting change in model outputs. This setup allows us to test whether information encoded at a given layer contributes to the final decision.

Our experiments reveal a consistent gap between probing performance and causal influence. Layers with highly separable representations often produce limited changes under intervention. This pattern appears across architectures. We also observe a clear asymmetry in vulnerability representations: replacing vulnerable activations with safe ones shifts predictions, but the reverse direction has little effect. This suggests that vulnerability signals act as localized features that need to be preserved, while safe

behavior is more stable and distributed.

To further examine representation structure, we complement the intervention analysis with CKA (Kornblith et al., 2019) and code retrieval experiments. The results show that downstream performance depends not only on whether information is present, but also on whether it is aligned with the representation space used by the task.

Overall, representation quality and causal importance do not always coincide. Understanding code models requires analyzing not only what information is encoded, but also how it is used during prediction.

Our main contributions are summarized as follows:

- We show that high probing performance does not necessarily imply strong causal influence, and that probing can overestimate the functional role of intermediate representations in code models.
- We identify a consistent asymmetry in vulnerability representations across architectures: removing vulnerability-related activations affects predictions, while injecting them into safe code has limited impact.
- We provide a unified analysis combining probing, activation patching, CKA, and retrieval evaluation, offering a layer-wise view of how code models organize and use task-relevant information.

2 Related Work

2.1 Probing and Representation Analysis

Probing methods are widely used to study whether neural representations encode specific linguistic or semantic properties. Early work introduced auxiliary classifiers to evaluate whether hidden states contain recoverable structural or semantic information (Shi et al., 2016; Adi et al., 2017; Belinkov et al., 2017). Subsequent studies applied probing to Transformer models and showed that intermediate layers often encode rich syntactic and semantic features that can be extracted using simple linear classifiers (Alain and Bengio, 2018; Conneau et al., 2018; Hewitt and Manning, 2019). Similar observations have also been reported for large language models and code models (Wang et al., 2026).

Several studies further examined how representations evolve across layers. Jawahar et al. (2019) and Rogers et al. (2020) found that lower layers capture lexical information, intermediate layers encode richer semantics, and upper layers become more task-specific. Representation similarity methods have also been widely used to analyze layer-wise behavior. In particular, centered kernel alignment (CKA) is a common tool for comparing representations across layers and models (Kornblith et al., 2019). Related work shows that representations become increasingly anisotropic and task-dependent in deeper layers (Ethayarajh, 2019).

Despite these insights, later work pointed out key limitations. Hewitt and Liang (2019) showed that strong probing performance can reflect probe capacity rather than representation quality. Elazar et al. (2021) demonstrated that information detectable by probing is not necessarily used during inference. More generally, probing measures extractability rather than functional reliance (Hewitt et al., 2021; Belinkov, 2022).

Our work builds on this line of research but focuses on the gap between extractability and causal usage. Instead of only testing whether vulnerability-related information is present, we directly examine whether intermediate representations affect model predictions through intervention.

2.2 Causal and Mechanistic Interpretability

Recent work on mechanistic interpretability studies how Transformer models process and route information during inference. Early analyses examined how attention heads and feed-forward layers contribute to specific behaviors (Elhage et al., 2021), shifting interpretability research from correlational analysis toward intervention-based approaches.

Several studies introduced causal methods to identify representations that directly contribute to model outputs. Vig et al. (2020) applied causal mediation analysis to study bias in language models. Meng et al. (2022) proposed activation patching to locate factual associations by replacing intermediate activations during inference. Geva et al. (2023) further analyzed how autoregressive models retrieve factual knowledge through specific components. These results show that intervention-based methods provide stronger evidence of functional usage.

Most existing work focuses on natural language models. Code models have received less attention, especially in security-related tasks such as vulnerability detection. Compared with natural language, source code has stricter structure and stronger long-

range dependencies, which may lead to different representation and usage patterns.

We conduct intervention-based analysis on both encoder-only and decoder-only code models to study the functional role of vulnerability-related representations.

2.3 Code Language Models and Vulnerability Detection

Pre-trained code language models achieve strong performance on tasks such as code search, vulnerability detection, and code generation (Husain et al., 2020; Fan et al., 2020; Chen et al., 2023). These models are typically based on Transformer architectures and trained on large-scale code corpora (Feng et al., 2020; Guo et al., 2021; Wang et al., 2021; Rozière et al., 2024; Guo et al., 2024). Compared with natural language, code exhibits stricter syntax, hierarchical structure, and long-range dependencies, making it a useful setting for studying internal representations.

Most prior work focuses on improving downstream performance through scaling, architectural changes, or task-specific training. Fewer studies examine how vulnerability-related semantics are represented and used during inference. Existing interpretability work on code models mainly relies on attention visualization, probing, or similarity analysis, which provide limited evidence about causal usage.

Our work addresses this gap by combining representation analysis with causal intervention. By comparing encoder-only and decoder-only models, we study not only where vulnerability-related information appears, but also whether it influences predictions.

3 Method

3.1 Representations

Consider a Transformer with L layers. We denote the hidden representation at layer l as $h_l \in \mathbf{R}^d$. For encoder-only models, we extract representations from the classification token. For decoder-only models, we use the hidden state at the final prediction position.

We distinguish between two representation types. *Intrinsic representations* are taken directly from intermediate layers, while *aligned representations* refer to representations after task-specific projection heads.

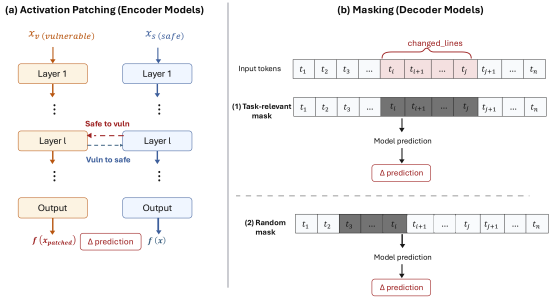

Figure 1: Overview of intervention methods. Left: activation patching replaces intermediate representations at layer l using paired samples in encoder models. Right: masking removes information from either task-relevant regions (changed lines) or random regions in decoder models. Prediction changes are used to measure the effect of interventions.

3.2 Probing

We use linear probing to assess whether task-relevant information is linearly accessible in each layer. For each h_l, we train a logistic regression classifier on frozen representations. Probing reflects how easily features can be extracted, but does not indicate whether the model relies on them during prediction.

3.3 Activation Interventions

To examine whether intermediate representations influence model predictions, we perform intervention-based analysis on vulnerable and fixed code samples. For encoder-only models, we apply activation patching by replacing representations at a selected layer with those from a paired sample, while keeping the rest of the forward pass unchanged. For decoder-only models, where representations evolve autoregressively, we instead apply masking-based interventions on task-relevant regions.

We measure the intervention effect as the signed change in the vulnerability prediction logit:

$$\Delta_l = f(x) - f(x^{(l)}_{\text{intervened}}) \quad (1)$$

where $x^{(l)}_{\text{intervened}}$ is the input after applying the intervention at layer l. Under this convention, a positive Δ_l indicates that the intervention reduces the vulnerability logit, whereas a negative Δ_l indicates that the intervention increases the vulnerability logit.

We compare interventions on paired samples with interventions on unrelated samples.

An illustration of this procedure is shown in Figure 1.

3.4 Representation Similarity

We use Centered Kernel Alignment (CKA) to measure similarity between representations across layers. By comparing representations pairwise, we examine how they evolve across depth and whether different layers exhibit distinct structural patterns.

4 Evaluation

4.1 Experimental Setup

Models. We evaluate two encoder-only models (CodeBERT (Feng et al., 2020), GraphCodeBERT (Guo et al., 2021)) and two decoder-only models (DeepSeek-Coder (Guo et al., 2024), CodeLlama (Rozière et al., 2024)).

Datasets. For vulnerability detection, we use the BigVul (Fan et al., 2020) dataset, which provides vulnerable and fixed code samples. We use the standard train/val/test split with 150,908/18,863/18,865 examples. For code retrieval, we use CodeSearchNet (Husain et al., 2020) and follow its official split. All datasets and pre-trained models used in this work are used solely for scientific research and evaluation purposes in accordance with their original licenses and intended use conditions.

Representation Extraction. For encoder-only models, we use the [CLS] representation at each layer, while for decoder-only models, we use the hidden state at the final position.

Probing. For each layer, we train a logistic regression classifier on frozen representations. For vulnerability detection, we report F1-Score, ROC-AUC and PR-AUC. For retrieval, we use cosine similarity and report MRR, Recall@1, and Recall@10.

Activation Interventions. We perform activation interventions on paired samples from BigVul. We compare interventions on paired samples with corresponding interventions on unrelated samples. We report the change in prediction logits and evaluate significance using Welch's t-test and the Mann–Whitney U test, along with Cohen's d.

Implementation Details. Experiments are conducted on a single NVIDIA A800 GPU. All implementations are based on PyTorch and HuggingFace Transformers. We use task-specific fine-tuned checkpoints for all models, with default hyperparameter settings. No additional preprocessing beyond tokenization is applied.

4.2 Layer-wise Representation Dynamics

We first examine how vulnerability-related information evolves across layers in different architectures using layer-wise probing on the BigVul dataset. Figure 2 presents the layer-wise probing performance measured by PR-AUC on the BigVul dataset. Across all architectures, task-relevant information emerges at relatively shallow layers, indicating that pre-trained code models can establish an initial distinction between vulnerable and non-vulnerable code early in the network.

Encoder-only models exhibit relatively concentrated representation formation. In CodeBERT and GraphCodeBERT, performance rises rapidly in the middle layers and saturates within a narrow depth range, suggesting that vulnerability-related semantics become linearly separable early through bidirectional contextual encoding.

Decoder-only models show a different layer-wise trend. DeepSeek-Coder develops initial vulnerability discrimination ability in shallow layers, where PR-AUC rises substantially during the early stages of the network. Unlike encoder-only models, whose performance rapidly concentrates within a narrow middle-layer region, decoder-only models exhibit a more gradual transition before reaching a stable plateau. Most performance gains are still formed within the first half of the network, but the increase is distributed over a broader range of layers rather than a single concentrated stage.

This difference indicates that vulnerability-related information is organized differently across architectures. In encoder-only models, task-relevant features become concentrated within a relatively localized range of middle layers. In contrast, decoder-only models distribute representation refinement over a wider depth range, leading to a smoother performance trajectory rather than a sharp mid-layer transition.

Although probing results reveal where task-relevant information becomes linearly accessible, they do not necessarily indicate whether the model causally relies on these representations during inference. Therefore, in Section 4.4, the causal role played by intermediate activations will be investigated through Activation Patching.

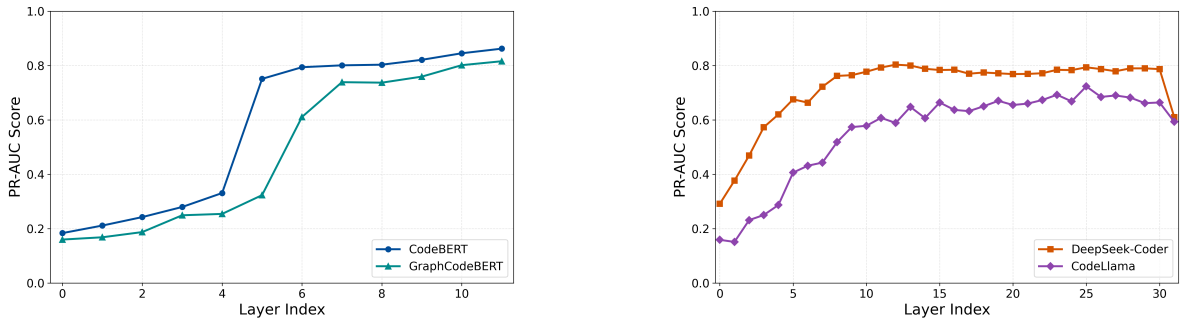

Figure 2: Layer-wise probing performance (PR-AUC) across architectures. Encoder-only models exhibit rapid mid-layer performance concentration, while decoder-only models show a broader and more gradual improvement pattern.

4.3 Representation Geometry and Task Alignment

4.3.1 Geometry of Vulnerability Representations

To further understand the layer-wise behaviors observed in probing experiments, we analyze representation similarity across layers using Centered Kernel Alignment (CKA). This analysis focuses on how representations evolve with depth and whether different architectures exhibit distinct patterns of representation organization.

The CKA results reveal clear differences between encoder-only and decoder-only models. In encoder-only models, representation similarity gradually decreases as layer distance increases. Neighboring layers remain highly similar, while shallow and deep layers become substantially less aligned. This pattern suggests that representations are continuously reorganized throughout the network depth rather than remaining geometrically stable. Compared with CodeBERT, GraphCodeBERT preserves stronger similarity across distant layers. The similarity between early and late layers remains relatively high, indicating that the incorporation of structural information leads to smoother representation transitions across depth. These observations are consistent with the probing results in Section 4.2, where encoder-only models exhibit rapid performance improvements within a relatively concentrated range of intermediate layers.

Decoder-only models exhibit a different pattern. In both DeepSeek-Coder and CodeLlama, large groups of neighboring layers remain highly similar across the early and intermediate portions of the network. Compared with encoder-only models, representation changes occur more gradually with depth. This behavior also matches the probing trajectories observed previously. Although shallow layers already establish initial vulnerability discrimination ability, later layers continue to refine these representations before performance stabilizes.

Another phenomenon appears at the final layer of decoder-only models. The similarity between the final layer and preceding layers decreases sharply compared with the relatively stable intermediate regions. The same layers also correspond to slight drops in probing performance. This suggests that the final layer becomes increasingly specialized toward the autoregressive prediction objective, which may weaken the linear separability of vulnerability-related features preserved in earlier layers.

4.3.2 Intrinsic vs Aligned Representations in Retrieval

We further examine how representation geometry changes under retrieval settings by comparing intrinsic representations with task-aligned representations on CodeSearchNet.

The results reveal a substantial difference between zero-shot retrieval and aligned retrieval, especially for decoder-only models. In the zero-shot setting, both DeepSeek-Coder and CodeLlama exhibit a pronounced non-monotonic trajectory across layers. Although shallow layers retain moderate retrieval capability, performance drops sharply in the early and intermediate layers before recovering near the top of the network.

For DeepSeek-Coder, MRR decreases from 0.359 at Layer 0 to below 0.05 across several intermediate layers, before rising rapidly near the final layers and reaching 0.626 at Layer 30. CodeLlama shows a similar pattern, where retrieval performance remains low across a large portion of the

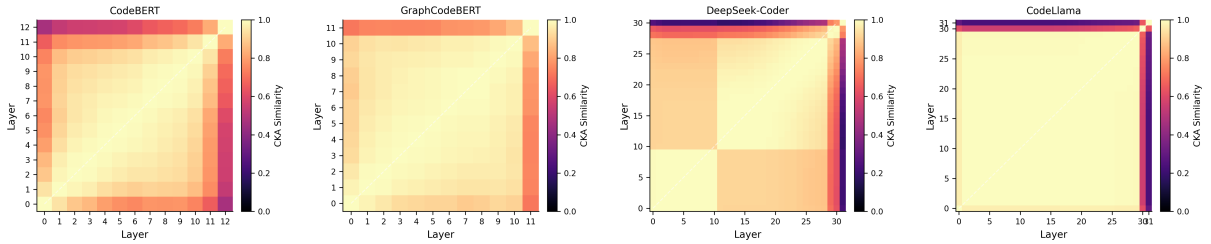

Figure 3: Layer-wise representation similarity measured by CKA across architectures.

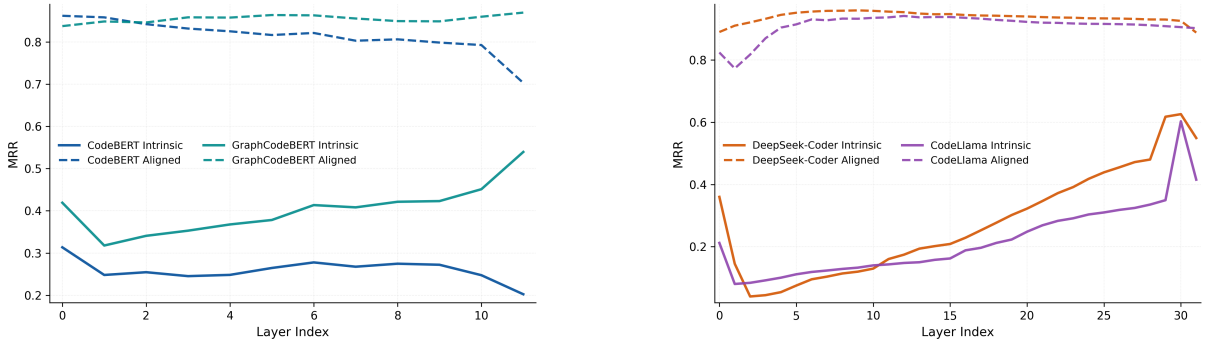

Figure 4: Layer-wise retrieval performance on CodeSearchNet under intrinsic (zero-shot) and projection-aligned settings.

network before increasing substantially near the top layers. These results indicate that intermediate representations in decoder-only models are not naturally aligned with the retrieval objective under zero-shot evaluation, even though later layers partially recover this alignment.

The aligned setting produces a substantially different pattern. After introducing a learned projection layer, the strongest retrieval performance no longer appears near the final layers. Instead, intermediate layers become the most effective retrieval representations. DeepSeek-Coder achieves its highest MRR at Layer 9, while CodeLlama peaks around Layer 12. Both models substantially outperform their zero-shot counterparts under this setting.

This contrast between intrinsic and aligned representations suggests that intermediate layers already contain highly informative semantic features for retrieval, but these features are not directly organized in a geometry suitable for similarity-based matching. The projection layer largely resolves this mismatch by mapping representations into a task-aligned embedding space.

Encoder-only models exhibit a different layer-wise pattern. In CodeBERT, aligned retrieval performance is strongest at shallow layers and gradually decreases with depth. This behavior suggests that deeper layers become increasingly specialized toward the pre-training objective, which weakens their direct suitability for retrieval.

GraphCodeBERT behaves differently from CodeBERT in both zero-shot and aligned settings. Retrieval performance improves more consistently with depth and reaches its best performance near the top layers under both evaluation protocols. Compared with CodeBERT, the layer-wise degradation is substantially weaker. This observation suggests that structural information helps preserve semantic consistency across deeper layers, making later representations more suitable for cross-modal alignment.

4.4 Causal Effects of Intermediate Representations

Linear probing identifies where task-relevant features become linearly accessible but does not establish whether these features are used during inference. To examine their functional role, we perform activation interventions on correctly classified samples and measure the resulting prediction shifts.

4.4.1 CodeBERT: Localized Activation Patching

We first analyze the encoder-only CodeBERT using localized activation patching on the *changed_lines* region.

Figure 5 shows the distribution of prediction

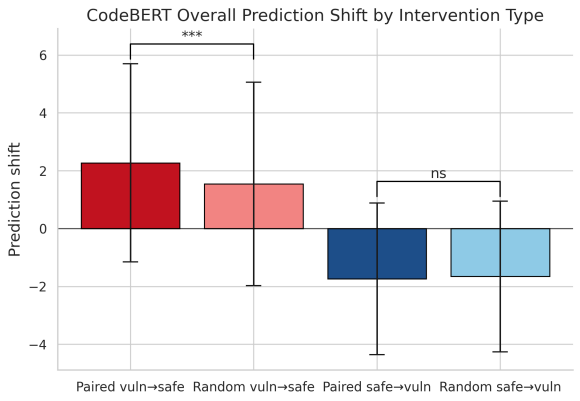

Figure 5: Global prediction shifts under different activation patching settings in CodeBERT.

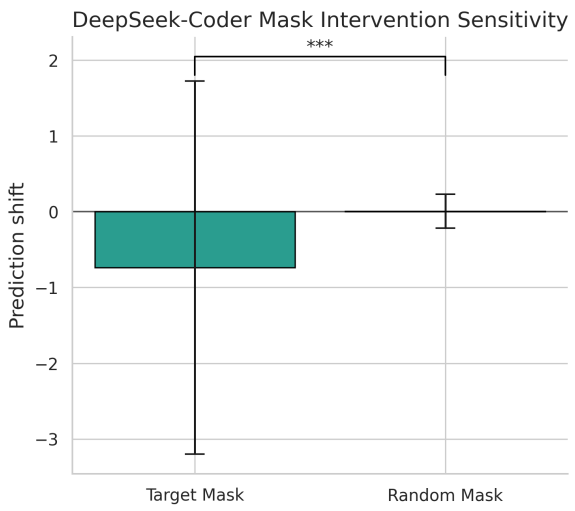

Figure 6: Prediction shifts under target-region masking and random masking in DeepSeek-Coder.

shifts under paired and random interventions.

In the *vuln-to-safe* direction, paired substitutions produce larger shifts than random substitutions (mean: 2.27 vs. 1.55). The difference is statistically significant (Welch's $p = 3.21 \times 10^{-4}$, Mann–Whitney $p = 6.79 \times 10^{-4}$, Cohen's $d = 0.208$), indicating that semantically aligned perturbations lead to a statistically reliable but moderate increase in prediction shift compared to random substitutions. In contrast, the *safe-to-vuln* direction does not show a statistically significant difference between paired and random interventions. As illustrated in Figure 5, the distributions of paired and random interventions largely overlap, with mean shifts of -1.74 and -1.66 and non-significant test results (Welch's $p = 0.588$, Mann–Whitney $p = 0.109$, Cohen's $d = -0.031$).

We observe an asymmetry between the two directions. Removing vulnerability-related features changes the predictions, while adding them has little effect.

4.4.2 DeepSeek-Coder: Mask-Based Intervention

We next evaluate the decoder-only DeepSeek-Coder using a masking-based intervention.

Figure 6 compares prediction shifts when masking task-relevant regions, identified as the changed lines in paired samples, against masking random regions of equal length.

Masking task-relevant regions consistently results in larger prediction shifts compared to random masking. The difference is highly significant (Welch's $p < 10^{-100}$, Mann–Whitney $p < 10^{-100}$) with a moderate effect size (Cohen's $d = -0.424$), indicating that the model is consistently sensitive to perturbations in task-relevant regions.

Compared to CodeBERT, the effect is more uniform across samples, suggesting that causal signals are more distributed rather than concentrated in a specific direction.

4.4.3 Summary

Across experiments, interventions on task-relevant locations consistently produce larger prediction shifts than those on random locations. This pattern appears in both encoder-only and decoder-only models, although it manifests differently across architectures. In CodeBERT, the effect is asymmetric: replacing vulnerable activations with safe ones leads to noticeable shifts, while the reverse intervention has a much smaller impact. In DeepSeek-Coder, a similar trend is observed under masking, where removing information from task-relevant regions leads to larger changes than masking random regions.

These results suggest that the impact of interventions depends on both the model architecture and how the intervention is applied. Additional results for GraphCodeBERT and CodeLlama are provided in Appendix B, showing similar trends across encoder-only and decoder-only models.

4.5 The Gap Between Extractability and Causal Usage

We compare intervention results with the probing analysis in Section 4.2.

A mismatch emerges in the layer-wise analysis. As shown in Figure 7, probing performance increases with depth, while prediction shifts remain relatively stable across layers. Interventions

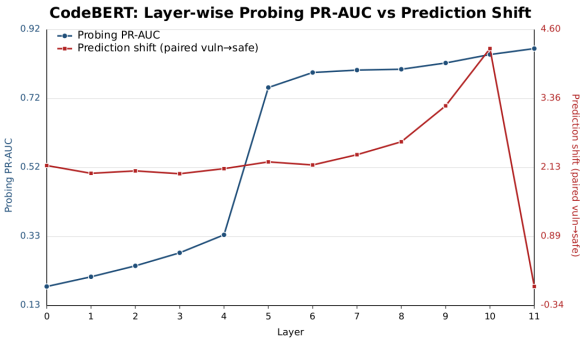

Figure 7: Layer-wise comparison between probing performance (PR-AUC) and prediction shifts under activation patching in CodeBERT.

in early layers already produce noticeable changes, and higher probing scores do not consistently correspond to larger shifts. Similar trends are observed in other models.

The effect also depends on intervention direction. In the *vuln-to-safe* setting, paired substitutions lead to larger shifts than random ones, whereas in the *safe-to-vuln* setting the effect is much weaker, even though vulnerable and safe samples are well separated under probing.

These results highlight a gap between what can be extracted from representations and how the model responds to interventions on them.

5 Discussion

Our results point to a gap between the presence of vulnerability-related information in representations and its role in model predictions. While probing shows that such information is broadly accessible across layers, intervention results indicate that its effect depends on where and how it is accessed.

One consistent finding is the importance of spatial alignment. In encoder-only models such as CodeBERT, interventions applied to the changed lines lead to clear prediction shifts, whereas random interventions have much weaker effects. This suggests that model predictions are more sensitive to representations in task-relevant regions than to globally available features.

We also observe an asymmetry in encoder-only models. In CodeBERT, removing vulnerability-related signals leads to noticeable changes in prediction, while introducing such signals has a much smaller effect, even though vulnerable and safe samples are well separated under probing. GraphCodeBERT shows a similar but less pronounced pattern.

In decoder-only models such as DeepSeek-Coder, masking task-relevant regions consistently produces larger prediction changes than masking random regions. This effect is not directional, but depends on whether relevant information is removed, indicating a different mode of dependence on input representations.

These patterns are further supported by label flip rate (LFR), which measures the proportion of samples whose predicted label changes after intervention. Across models, interventions on task-relevant regions lead to substantially higher flip rates than those on random regions. In CodeBERT, paired interventions on changed lines yield higher flip rates than random substitutions in the *vuln-to-safe* setting, while the effect is much weaker in the reverse direction. DeepSeek-Coder shows a large gap between target masking and random masking, consistent with the strong region-based effect observed in logit shifts.

Taken together, these results show that probing performance alone does not fully characterize how representations are used. Features that are easily extracted may have limited impact on predictions unless interventions are aligned with task-relevant regions, and their effects can vary depending on how the intervention is applied.

6 Conclusion

In this work, we study how vulnerability-related information is represented and used in pre-trained code models. Combining layer-wise probing and intervention-based analysis, we examine both the extractability and functional role of intermediate representations.

We find that vulnerability-related features are broadly accessible across layers, but their impact on predictions is not uniform. Instead, it depends on spatial alignment, intervention direction, and model architecture. In encoder-only models, effects are localized and asymmetric, while decoder-only models show more uniform sensitivity to masking task-relevant regions.

These results suggest that extractable representations do not necessarily correspond to their use in prediction. Evaluating model behavior therefore requires going beyond probing and incorporating intervention-based analysis. Future work can extend this framework to other tasks and more fine-grained interventions.

Limitations

Our analysis is based on a limited set of models and tasks. Although we include both encoder-only and decoder-only architectures, the observations in this paper may not directly extend to other model families or training settings.

The intervention results should also be interpreted with some caution. Activation patching provides a controlled way to modify intermediate representations, but it only captures part of the computation. In particular, it does not account for more complex interactions across layers.

In addition, our experiments focus on vulnerability detection and code retrieval. It remains unclear whether the same patterns would appear in other software engineering tasks.

Finally, while we study how representations affect model predictions, this does not imply that the models themselves make reliable security judgments. The results are intended to describe model behavior rather than to evaluate their practical reliability.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *International Conference on Learning Representations*.

Guillaume Alain and Yoshua Bengio. 2018. Understanding intermediate layers using linear classifier probes. *Preprint*, arXiv:1610.01644.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872, Vancouver, Canada. Association for Computational Linguistics.

Yizheng Chen, Zhoujie Ding, Lamya Alowain, Xinyun Chen, and David Wagner. 2023. Diversevul: A new vulnerable source code dataset for deep learning based vulnerability detection. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*, RAID '23, page 654–668, New York, NY, USA. Association for Computing Machinery.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. 2021. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, and 6 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China. Association for Computational Linguistics.

Jiahao Fan, Yi Li, Shaohua Wang, and Tien N. Nguyen. 2020. A c/c++ code vulnerability dataset with code changes and cve summaries. In *Proceedings of the 17th International Conference on Mining Software Repositories*, MSR '20, page 508–512, New York, NY, USA. Association for Computing Machinery.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547, Online. Association for Computational Linguistics.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235,

Singapore. Association for Computational Linguistics.

Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, Shujie LIU, Long Zhou, Nan Duan, Alexey Svyatkovskiy, Shengyu Fu, Michele Tufano, Shao Kun Deng, Colin Clement, Dawn Drain, Neel Sundaresan, Jian Yin, Daxin Jiang, and Ming Zhou. 2021. Graphcode{bert}: Pre-training code representations with data flow. In *International Conference on Learning Representations*.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. 2024. Deepseek-coder: When the large language model meets programming – the rise of code intelligence. *Preprint*, arXiv:2401.14196.

John Hewitt, Kawin Ethayarajh, Percy Liang, and Christopher Manning. 2021. Conditional probing: measuring usable information beyond a baseline. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1626–1639, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2020. Codesearchnet challenge: Evaluating the state of semantic code search. *Preprint*, arXiv:1909.09436.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–3529. PMLR.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, and 7 others. 2024. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Simas Sakenis, Jason Huang, Yaron Singer, and Stuart Shieber. 2020. Causal mediation analysis for interpreting neural nlp: The case of gender bias. *Preprint*, arXiv:2004.12265.

Chenglong Wang, Yifu Huo, Yang Gan, Yongyu Mu, Qiaozhi He, Murun Yang, Bei Li, Chunliang Zhang, Tongran Liu, Anxiang Ma, Zhengtao Yu, Jingbo Zhu, and Tong Xiao. 2026. Probing preference representations: A multi-dimensional evaluation and analysis method for reward models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(39):33404–33412.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. 2021. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8696–8708, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

A Full Layer-wise Probing Results

This section presents the complete layer-wise probing results for all evaluated models on the vulnerability detection task. In the main paper, we primarily report PR-AUC since it is more informative under class imbalance. Here we additionally

Model	Direction	Paired	Random
CodeBERT	vuln→safe	36.5	28.0
CodeBERT	safe→vuln	12.5	18.5
GraphCodeBERT	vuln→safe	46.3	32.8
GraphCodeBERT	safe→vuln	27.7	22.2

Table 1: Prediction flip rates (%) under paired activation patching.

Model	Target	Random	Last Token
DeepSeek-Coder	1.09	2.23	19.30
CodeLlama	0.52	0.56	0.17

Table 2: Prediction flip rates (%) under masking-based interventions in decoder-only models.

provide F1-Score and ROC-AUC curves to demonstrate that the observed layer-wise trends remain consistent across different evaluation metrics.

A.1 F1-Score Across Layers

Figure 8 shows the layer-wise F1-Score trajectories for all models.

A.2 ROC-AUC Across Layers

Figure 9 presents the corresponding ROC-AUC results. The overall trends are consistent with the PR-AUC observations discussed in Section 4.2.

B Probing Performance and Causal Effects

To further examine the relationship between probing performance and causal influence, we compare layer-wise probing scores with intervention-induced prediction shifts.

B.1 Encoder-Only Models

Figure 10 compares probing performance and activation-induced prediction shifts across layers in encoder-only models.

B.2 Decoder-Only Models

Figure 11 presents the corresponding results for decoder-only models.

C Prediction Flip Statistics

Tables 1 and 2 report prediction flip statistics under different intervention settings. Table 1 presents activation patching results on paired vulnerable–safe samples for encoder-only models. Table 2 reports masking-based interventions for decoder-only models. Rates are reported as percentages.

D Statistical Testing Details

Tables 3 and 4 summarize the statistical results for intervention experiments across different architectures.

Model	Target	Random	Welch p	MWU p	Cohen's d
CodeLlama	0.014	0.028	1.31×10^{-12}	1.57×10^{-46}	-0.126
DeepSeek-Coder	-0.737	0.005	4.85×10^{-122}	1.284×10^{-141}	-0.424

Table 3: Statistical test results for masking-based interventions in decoder-only models.

E Use of AI Tools

We used AI-assisted tools for language polishing and minor editing. All technical content and conclusions were developed by the authors.

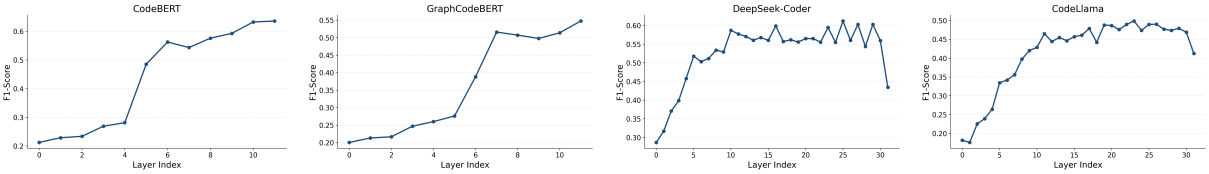

Figure 8: Layer-wise F1-Score across models on BigVul.

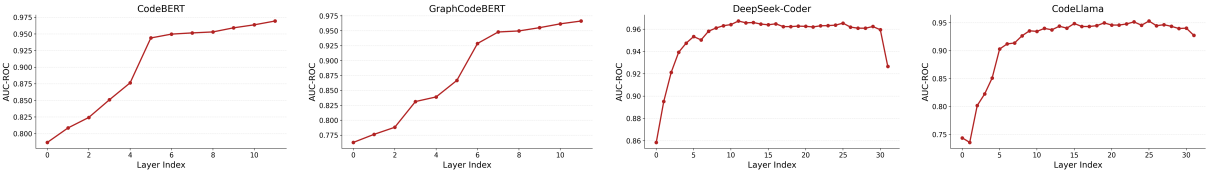

Figure 9: Layer-wise ROC-AUC across models on BigVul.

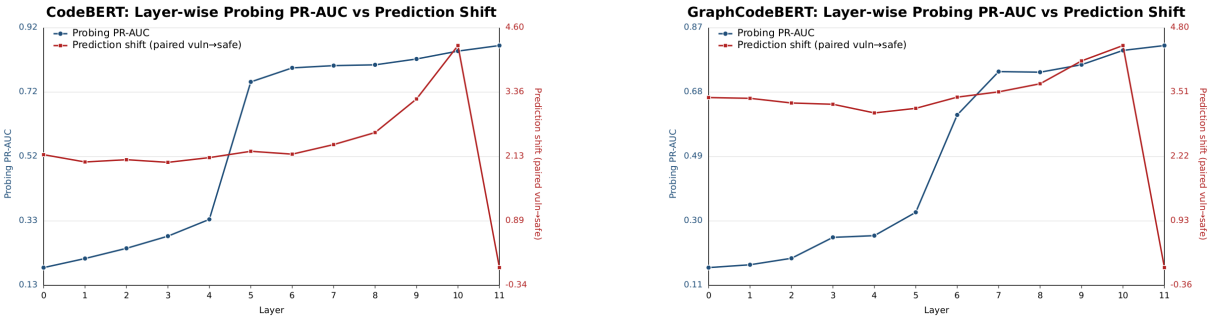

Figure 10: Layer-wise probing performance and prediction shifts in encoder-only models.

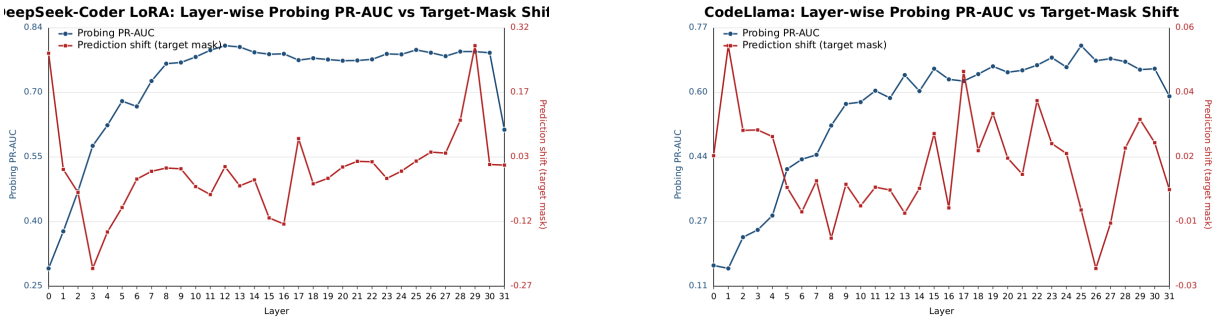

Figure 11: Layer-wise probing performance and prediction shifts in decoder-only models.

Model	Direction	Paired	Random	Welch p	MWU p	Cohen's d
CodeBERT	vuln→safe	2.27	1.55	3.21×10^{-4}	6.79×10^{-4}	0.208
CodeBERT	safe→vuln	-1.74	-1.66	0.588	0.109	-0.031
GraphCodeBERT	vuln→safe	3.24	2.50	8.07×10^{-5}	9.57×10^{-4}	0.228
GraphCodeBERT	safe→vuln	-2.78	-2.26	0.0018	9.15×10^{-6}	-0.180

Table 4: Statistical test results for paired activation patching in encoder-only models.