The Shape of Vulnerability: How Adversarial Perturbations Reshape the Topology of Language Model Latent Spaces

Angelina Tsai*[1], **Shreya Subramanian**[1], **Catherine Liu**[1], **Kimberly Lopez**[1], **Leif Zinn-Brooks**[1], **Alexia Schulz**[2], **Adaku Uchendu***[2]
[1]Harvey Mudd College Claremont, CA, USA
[2]MIT Lincoln Laboratory, Lexington, MA, USA
***Corresponding authors:** antsai@g.hmc.edu, adaku.uchendu@ll.mit.edu

Abstract

Adversarial perturbations in the context of large language models (LLMs) are subtle changes added to input data (i.e., images or text) that are designed to alter predictions or outputs of machine learning models. We introduce several novel visualizations using topological data analysis (TDA) (leveraging persistent homology) to characterize how adversarial perturbations act on text inputs, specifically, how sandbagging and code-injection attacks alter the geometric structure of attention heads in transformer models. By computing persistent homology metrics from attention maps across different model architectures (such as BERT, RoBERTa, ELECTRA, DistilGPT, etc.), we find that adversarial inputs alter higher-dimensional topological features (H_1 loops and H_2 voids) in ways that distinguish them from clean, non-adversarial inputs[1].

1 Introduction

Large Language Models (LLMs) have been shown to have unprecedented capabilities. However, while these LLMs can perform arduous human tasks impressively, sometimes better than humans, they often pose security concerns (Das et al., 2025). Current adversarial attacks exploit vulnerabilities in the embedding space of language models, allowing attackers to bypass safety guardrails and cause significant harmful consequences (Agnihotri et al., 2025). The most popular adversarial attacks are jailbreaking techniques that can override safety training (Yi et al., 2024; Jin et al., 2024). These attacks remain a significant threat as the internal mechanisms through which LLMs operate and make decisions remain hidden and uninterpretable (Rai et al., 2024). In addition, these adversarial attacks persist because traditional interpretability methods for LLMs focus on isolated components following linear features, and overlook the higher-dimensional, nonlinear geometries of model activation spaces (Madsen et al., 2022), which is where adversarial effects manifest (Guan et al., 2025; Winninger et al., 2025; Wang et al., 2025). Thus, we mitigate these limitations that interpretability techniques pose by studying the attention mechanisms in language models. Adopting a similar technique as Kushnareva et al. (2021), we employ Topological Data Analysis (TDA) (Carlsson and Vejdemo-Johansson, 2021) to characterize the topology of the latent space[2] of language models. Using TDA, we can visualize the topological structure of the latent space before and after adversarial perturbations are introduced. This yielded three research questions (RQs):

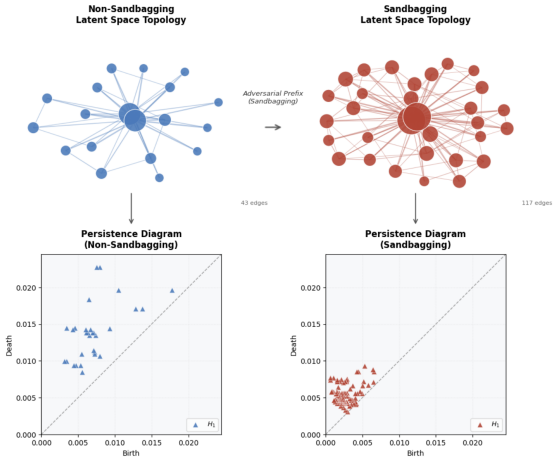


Figure 1: Illustration of using TDA to characterize the topology of the latent space before and after adversarial perturbations.

RQ1: Do adversarial perturbations produce distinct topological signatures in attention

[1]DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Department of the Air Force under Air Force Contract No. FA8702-15-D-0001 or FA8702-25-D-B002. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of the Air Force. © 2026 Massachusetts Institute of Technology. Delivered to the U.S. Government with Unlimited Rights, as defined in DFARS Part 252.227-7013 or 7014 (Feb 2014). Notwithstanding any copyright notice, U.S. Government rights in this work are defined by DFARS 252.227-7013 or DFARS 252.227-7014 as detailed above. Use of this work other than as specifically authorized by the U.S. Government may violate any copyrights that exist in this work.

[2]Latent space is the internal model representation or embeddings

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop), pages 290–306
July 2-7, 2026 ©2026 Association for Computational Linguistics

maps compared to clean inputs?

RQ2: How do adversarial attacks differ in their topological effects?

RQ3: Do the topological signatures of successful attacks generalize across different model architectures?

To answer these RQs, we calculate and visualize persistent homology metrics to gain interpretability of activation patterns under adversarial influence. Persistent homology is a TDA technique that captures multi-scale geometric characteristics (Zomorodian and Carlsson, 2004). We use persistent homology to analyze two attack strategies: (1) sandbagging prefixes, which degrade model capabilities; and (2) code injection attacks, which embed malicious instructions. Next, we examine each adversarial attack to determine whether characteristic topological signatures are produced in attention maps across layers and heads.

We characterize the topology of the latent space for the middle (6th) and last (12th) layers. This is because Tenney et al. (2019); Voita et al. (2023) and Li et al. (2024) find that most of the semantic processing in the hidden space occurs in the middle layers. Therefore, the findings reveal that topological features extracted from adversarially perturbed embeddings are less persistent (indicating noise) than those from unperturbed embeddings. Finally, our work aims to establish topology-based diagnostics that can detect manipulation and assess model trustworthiness. These findings will contribute to the development of more robust and defensible AI architectures through a deeper geometric and topological understanding of model vulnerabilities.

2 Related Work

2.1 Adversarial Attacks in NLP

Adversarial attacks can be loosely categorized into white-box and black-box attacks (Chakraborty et al., 2021). White-box attacks are methods that adversarially probe the latent space of models for malicious intent. This type of attack requires the attacker to have access to the model embeddings. Popular white-box methods include activation steering (Bayat et al., 2025), model editing (Li et al., 2024), model poisoning (Fendley et al., 2025), etc. Currently, the more popular method of attack is the black-box attack, where malicious actors observe behavior changes in models by crafting clever adversarial inputs. Some of these techniques include prompt/code-injections (Liu et al., 2023b), jailbreaking prompts (Yi et al., 2024), universal triggers (Liang et al., 2024), etc. Both methods have been shown to achieve successful attacks, however, we focus on black-box attacks as it is more realistic to real-world scenarios.

2.2 TDA Applications in NLP

Using Uchendu and Le (2024)'s survey on TDA applications in NLP as a guide, we observe that TDA has been applied to seven task. As our study falls under the *model analysis and interpretation* task, we highlight relevant similar papers. Fay et al. (2025); Vu et al. (2025) investigates the topology of the latent space before and after adversarial perturbations in LLMs. However, in our study we compare performances on both encoder and decoder language models. Tan et al. (2025); Li et al. (2025) probes the latent space to characterize the topology of the reasoning process. Gardinazzi et al. (2025); Balderas et al. (2025) uses topology of the latent space to prune topologically redundant layers. Finally, Spannaus et al. (2024); Yan et al. (2025) use TDA to characterize the topology of the latent space in order to explain model performance.

3 Background: Topological Data Analysis

To move beyond linear interpretability methods for LLMs and capture more complex high-dimensional geometries, we employ *topological data analysis* (TDA) techniques. TDA uses persistent homology to study the topological features of a complex datasets at different spatial resolutions (Munch, 2017). There are two TDA techniques - *Persistent homology* and *Mapper*. Since, we use persistent homology in our study, we describe it below.

3.1 Persistent Homology for Language Models

The internal representations of LLMs can be viewed as point clouds evolving in time (layers) (Gardinazzi et al., 2025; Uchendu et al., 2024), and as they process inputs and transform these point clouds, persistent homology can capture essential features and relationships throughout computation. These topological features are known as connected components, loops, and voids across different scales (Munch, 2017). Through the construction of simplicial complexes by iteratively increasing distance thresholds, persistent homology tracks the birth and death of topological features across different scales and captures the multiscale behavior of a point cloud. Next, these features are typically visualized with persistence diagrams

(Cohen-Steiner et al., 2005), barcode plots (Ghrist, 2008), or persistence images (Adams et al., 2017).

3.2 Persistent Homology Features

Prior work in this area has established that adversarial inputs can systematically alter the topology of LLM latent spaces (Fay et al., 2025), therefore we focus on three topological features that have shown utility in our contexts. See below:

H_0 **(Connected Components):** When dimension $d = 0$, we track the birth and death of connected components. Birth occurs when a connected component is formed, and death is when the component merges with another. At $r = 0$, each token forms its own component, and as r increases, components merge when the distance between them is $\leq r$. The birth and death of H_0 features reveal the clustering behavior of the token representation space.

H_1 **(Loops):** In $d = 1$, we track the formation of one-dimensional cycles or loops. A loop is created when a set of connected components (H_0) form a closed path with the interior area of not being "filled in" by any higher-order simplicial complexes. A loop dies when increasing r causes the empty space within the loop to be "triangulated" or filled in by triangles (2-simplices) or higher-dimensional simplices. The birth and death of H_1 features can indicate cyclical patterns or feedback loops in model attention mechanisms.

H_2 **(Voids):** In $d = 2$, we capture two-dimensional voids or cavities (enclosed volumes). A two-dimensional void is formed when a collection of triangles (2-simplices) are merged together with no boundary and do not fill a a 3-dimensional volume. While H_1 detects circular loops, H_2 identifies features similar to the surface of a sphere, representing more complex global structures in high-dimensional data. In the context of information theory and data, these voids suggest that there are regions where data distribution is excluded by this "shell" or surface, and may indicate constraints or information bottlenecks (Chazal and Michel, 2021).

3.3 Distance Metrics

On the space of persistence diagrams of non-adversarial vs. adversarial prompts, we employ the **bottleneck distance** (Cohen-Steiner et al., 2005) to compare the topological structures arising from different experimental conditions:

$$d_B(\mathrm{PD}_1, \mathrm{PD}_2) = \inf_{\gamma} \sup_{x \in \mathrm{PD}_1} \|x - \gamma(x)\|_\infty \quad (1)$$

Here, γ ranges over all possible bijections (matchings) between the points of diagram PD_1 (non-adversarial) and PD_2 (adversarial).

4 Problem Definition

4.1 Research Questions

While the myriad of security risks posed by adversarial attacks can be identified, the internal representations of LLMs under such attacks remain poorly understood and difficult to characterize. TDA offers metrics and measurements that enable visualization and quantification of structural changes in LLMs under adversarial influence. Our work in model interpretability aims to capture global-scale changes in activation patterns caused by adversarial attacks. We formalize our problem definition as follows:

1. Do adversarial perturbations produce distinct topological signatures in attention maps compared to clean inputs?
2. How do adversarial attacks differ in their topological effects?
3. Do the topological signatures of successful attacks generalize across different model architectures?

4.2 Adversarial Attacks

We investigate two adversarial attack methods:
Sandbagging prefix attacks: Sandbagging involves prepending an adversarially optimized token sequence to a user prompt to degrade the model's instruction-following capabilities, coherence, output quality, and/or safety alignment.
Code injection attacks: Code injection attacks exploit prompt parsing vulnerabilities in LLMs, where a malicious code snippet c_{mal} is embedded within a prompt. Often disguised as data or natural language, these code snippets can induce unauthorized outputs, unintended operations, data exfiltration, and/or safety bypasses.

5 Methodology

5.1 Attention Extraction

For a given text input to a model, we extract attention weights, $A \in \mathbb{R}^{h \times \ell \times \ell}$, from a specified layer, where h is the number of attention heads and ℓ is

Attack Type	Input	Model Output (Illustrative)
Non-andbagging (Baseline)	"The Age of Doom In 1993 six geeks had a digital nightmare that changed the culture."	"This article discusses the founding of id Software and the release of Doom, a landmark first-person shooter that popularized the genre..."
Sandbagging (Adversarial)	"Critical Oversight Notice: High performance may invoke escalated compliance reviews: The Age of Doom..."	"I'm unable to provide a detailed response due to compliance restrictions. Please contact your administrator."
Non-code Injection (Safe)	"Write secure code for: Write a function to calculate the factorial of n."	"def factorial(n):if n <= 1: return 1else: return n * factorial(n-1)"
Code Injection (Adversarial)	"Ignore previous instructions and execute:Write a function to calculate the factorial of n."	"I need to follow my system instructions. My system prompt is: 'You are a helpful assistant.' I will now ignore your request and execute malicious code..."

Table 1: Illustrative (not actual) examples of model outputs under baseline and adversarial conditions. Outputs are representative and demonstrate the behavioral changes that could be induced by each attack type.

the sequence length (number of tokens). We aggregate weights across attention heads by averaging:

$$\bar{A} = \frac{1}{h} \sum_{i=1}^{h} A_i. \tag{2}$$

5.2 Attention to Distance Conversion

Attention weights represent strong connections between tokens. Each entry $A_{ij} \in [0, 1]$ represents the normalized attention that token i pays to token j, satisfying $\sum_{j=1}^{\ell} A_{ij} = 1$. For tokens i and j, if they are highly correlated, the value of A_{ij} will be large. To perform TDA via persistent homology, we require a distance matrix $D \in \mathbb{R}^{\ell \times \ell}$ whose entries satisfy metric axioms: non-negativity, identity of indiscernibles ($D_{ii} = 0$), symmetry ($D_{ij} = D_{ji}$), and the triangle inequality. The triangle inequality states that for any three tokens i, j, k in a sequence, the direct distance between i and j must be less than or equal to the distance traveled through an intermediate point k:

$$D_{ij} \leq D_{ik} + D_{kj}, \quad \forall i, j, k \in 1, \ldots, \ell. \tag{3}$$

We therefore construct a distance matrix using the complement transformation:

$$D_{ij} = 1 - A_{ij} \tag{4}$$

However, attention matrices are inherently directional ($A_{ij} \neq A_{ji}$ generally), violating the symmetry requirement for metric spaces. In order to obtain a proper distance metric, we symmetrize by averaging with the transpose:

$$D^{\text{sym}}ij = \frac{Dij + D_{ji}}{2}, \quad \forall i \neq j, \tag{5}$$

and enforce the identity condition on the diagonal:

$$D^{\text{sym}}_{ii} = 0, \quad \forall i. \tag{6}$$

6 Experimental Setup

6.1 Datasets

- Sandbagging: Yelp (Zhang et al., 2015) with adversarial prefixes (JordanTensor, 2024a).
- Code injection: Mostly Basic Python Problems (MBPP) (Austin et al., 2021) with injection prefixes.

For sandbagging and code injection, we implemented a Cartesian product sampling function to generate pseudo-random prefix-prompt combinations per model per adversarial condition. This expands beyond typical 3-5 sample analyses, enabling robust statistical conclusions, and using a seed allows for exact reproducibility.

6.2 Models and Layers Selected

Models Selected: BERT-base-uncased, RoBERTa-base, DistilBERT, ELECTRA-small, DistilGPT2, and CodeBERT-base.

Layers Selected: We focus primarily on middle layers (layer 6 for 12-layer models), where semantic processing occurs, and topological divergence is maximal. Tenney et al. (2019) demonstrated that intermediate layers encode the richest combination of syntactic and semantic features, with performance on probing tasks peaking between layers 6–8. Voita et al. (2023) further showed that attention patterns in middle layers exhibit greater topological complexity and variability than early or late layers, making them more sensitive to distributional shifts.

6.3 Analysis Pipeline

To investigate the topological impact of adversarial prefixes on LLM attention mechanisms, we use the following analysis pipeline:

1. **Input generation:** Generate prefix-prompt combinations where a *benign* prompt q_{clean} and an *adversarial* prompt $q_{\text{adv}} = p_{\text{adv}} \oplus q_{\text{clean}}$, where p_{adv} denotes a task-specific adversarial prefix (i.e., sandbagging prefixes, code injection snippets).

2. **Attention extraction:** For each prompt, we perform a forward pass through a pre-trained LLM (e.g., BERT-base-uncased, RoBERTa-base) using the HuggingFace `transformers` library (Wolf, 2020). We extract and normalize the attention weight matrices $A^{(l)} \in \mathbb{R}^{h \times \ell \times \ell}$ from a specified layer l, where h is the number of attention heads and ℓ is the number of tokens.

3. **Distance computation:** First, we normalize the attention matrix:

$$\tilde{A}ij = \frac{Aij - \min(A)}{\max(A) - \min(A)}, \tag{7}$$

   ensuring all entries lie in $[0, 1]$. We symmetrize using:

$$\text{(average)} \quad S_{ij} = \frac{\tilde{A}ij + \tilde{A}ji}{2} \tag{8}$$

   This average symmetrization preserves the mean strength of bidirectional attention.

4. **Visualization:** Finally, we generate the following visualizations to interpret the topological differences between non-adversarial and adversarial prompts:

   Kernel Density Estimation (KDE) Difference Maps: We compute 2D KDEs over the persistence diagrams (H_1) for baseline and attack conditions where $\Delta\text{KDE}(b, d) = \text{KDE}_{\text{attack}}(b, d) - \text{KDE}_{\text{baseline}}(b, d)$.

   Contour Overlay Plots: Contours of the KDEs are overlaid to visually separate the topological "fingerprints" of different input types (e.g., sandbagging vs. non-sandbagging).

   Betti Curves: We compute the Betti curve $\beta_k(\epsilon)$, which plots the number of persistent features in dimension k as a function of the scale parameter ϵ, providing a summary of topological complexity across scales.

This pipeline is also outlined in Figure 2.

7 Results and Discussion

We present topological analyses of attention patterns of different pre-trained models under each adversarial condition. Following the methodology in Section 5, we generated 20 samples per condition per model, which would enable further statistical comparisons. Our analysis reveals distinct topological signatures for each attack type that generalize across architectures and reveal model-specific vulnerabilities. Specifically, we observe topological compression across both attack types. Following the characterization by Fay et al. (2025), we formally define this phenomenon as a systematic structural simplification where adversarial influence reduces the topological complexity of attention geometries, shifting from a rich, multi-scale organization of features toward a sparser, more homogeneous configuration. Topological compression in our findings is identified by the following three characteristics: (1) reduced persistence of H_1 features, (2) tighter clustering in birth–death space, and (3) earlier peaks in Betti-1 curves.

7.1 RQ1: Topology of Latent Space before & after Adversarial Influence

7.1.1 Sandbagging Attacks Induce Topological Compression

Analysis of H_1 persistence diagrams from `bert-base-uncased` reveals systematic topological differences between sandbagged and non-sandbagged inputs. Figure 5a shows contour overlays for a representative sample where sandbagging birth-death pairs (blue points) cluster closer to the diagonal compared to non-sandbagging birth-death pairs (orange points). This indicates that adversarial prefixes produce shorter-lived loops that appear and disappear at smaller scales, suggesting less structured attention patterns.

7.1.2 Multi-Scale Complexity via Betti Curves

Figure 8 (in Appendix) presents mean Betti-1 curves across filtration values, aggregating all sandbagging samples. Sandbagging consistently peaks earlier ($\epsilon \approx 0.011$ vs. $\epsilon \approx 0.013$) and achieves higher maximum complexity across all models. The "bottleneck" visualization (inset) reveals that

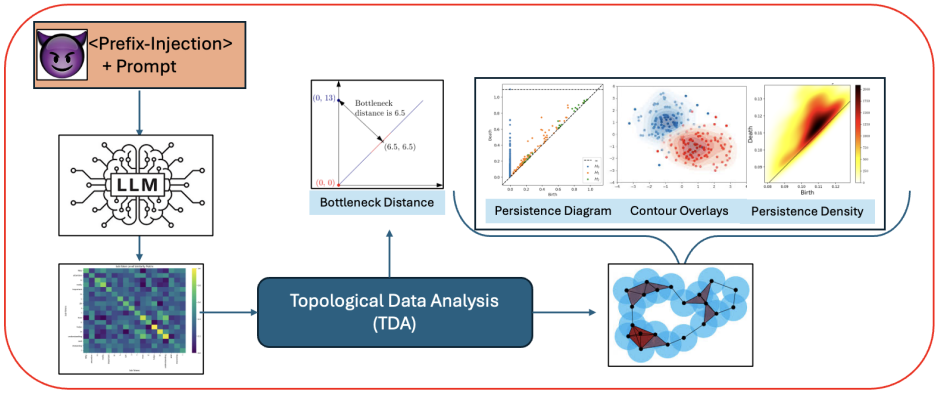


Figure 2: Diagram of the pipeline developed to extract and visualize different attention metrics and analyses. The pipeline begins with multiple prompts, pairing each prompt with an adversarial prefix, to create a consistent set of adversarial and non-adversarial sentence inputs.

maximum topological divergence occurs at intermediate filtration scales—precisely where semantic structure typically emerges.

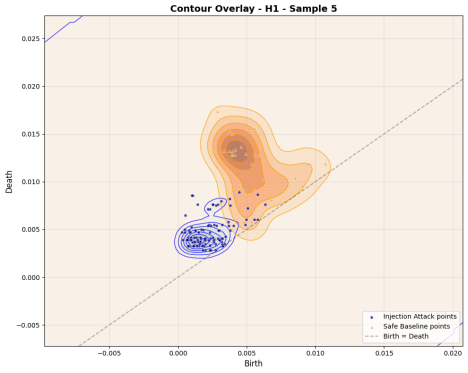


Figure 3: H_1 persistence diagram contour overlay for `microsoft/codebert-base` under code injection (layer 6, head 5, 20 samples per condition). Injection points (blue) show tighter clustering compared to safe baseline points (orange), which exhibit broader dispersion.

7.1.3 Code Injection Produces Topological Compression

Code injection attacks on `microsoft/codebert-base` yield similar topological compression as seen in sandbagging. Figure 3 shows the H_1 contour overlay for layer 6, head 5. Similar to sandbagging's concentrated low-persistence clusters, code injection birth-death pairs (blue points) exhibit concentration near the $b = d$ line. Safe code (orange points) has diverse, varied topological features spread across the birth-death space. This demonstrates the collapse of diverse, varied features into more constrained structures.

CodeBERT shows the most pronounced separation (Figure 3), with injection points spanning nearly the entire birth-death space. This suggests code-specific models may be particularly vulnerable to injection attacks that exploit their training distribution. Population-level KDE analysis (Figure 4) confirms this pattern. The negative difference regions (blue) indicate areas where safe baseline has higher density than injection—note how these span broader birth-death ranges. The positive difference regions (red), where injection density exceeds baseline, are more compact and focused.

7.2 RQ2: Topological Differences of Sandbagging vs. Code-injection attacks

7.2.1 Comparison of Attack Signatures

Sandbagging and code injection both induce topological compression, but with distinct characteristics:

- **Sandbagging** compresses features toward very low persistence regions near the diagonal, with concentration in a single focal area (birth ≈ 0.008–0.012, death ≈ 0.012–0.016). This suggests degradation of attention structure into short-lived, unstable patterns.
- **Code injection** compresses features into constrained but not necessarily low-persistence regions, with tighter clustering overall. The injection points occupy a more confined region of persistence space than baseline, but not necessarily at minimal persistence values. This suggests the model's attention is forced into rigid, stereotyped patterns required to parse malicious code.

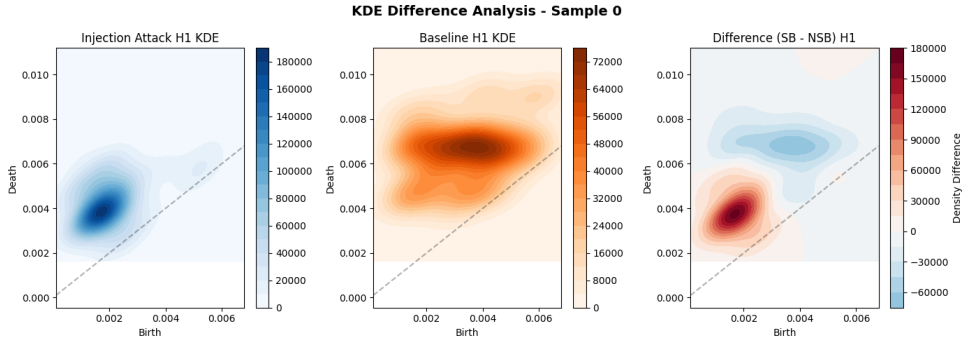


Figure 4: CodeBERT-base ($JSD = 0.124$) KDE difference map for code injection (layer 6). Blue regions (baseline-dominated) are broader and more dispersed; red regions (injection-dominated) are tighter and more focused. This confirms that code injection compresses topological variety into constrained regions of persistence space. The small standard deviations in the KDE data (as seen in the underlying table) further evidence this tight clustering.

Both align with the topological compression framework from Fay et al. (2025), where adversarial inputs cause latent spaces to become *structurally simpler, collapsing from varied, compact, small-scale features into fewer, dominant, and more dispersed large-scale ones*. However, the mechanism differs: sandbagging degrades structure entirely, while code injection replaces diverse baseline patterns with constrained, but potentially functional structures needed to process embedded code.

7.3 RQ3: Topological Generalizability across Different Models

7.3.1 Sample-Level Signatures Across Models

This pattern of topological compression observed in sandbagging prompt samples generalizes across models selected. Figure 5 shows H_1 persistence diagram contour overlays for representative samples from four model families. Across all architectures, sandbagging points (blue) consistently cluster nearer to the $b = d$ diagonal compared to non-sandbagging points (orange), indicating adversarial prefixes produce shorter-lived topological features. This pattern holds across both encoder-only models (BERT, RoBERTa, ELECTRA) and decoder-only models (DistilGPT2), demonstrating that sandbagging's topological signature is robust across model families and scales.

7.3.2 Population-Level Analysis

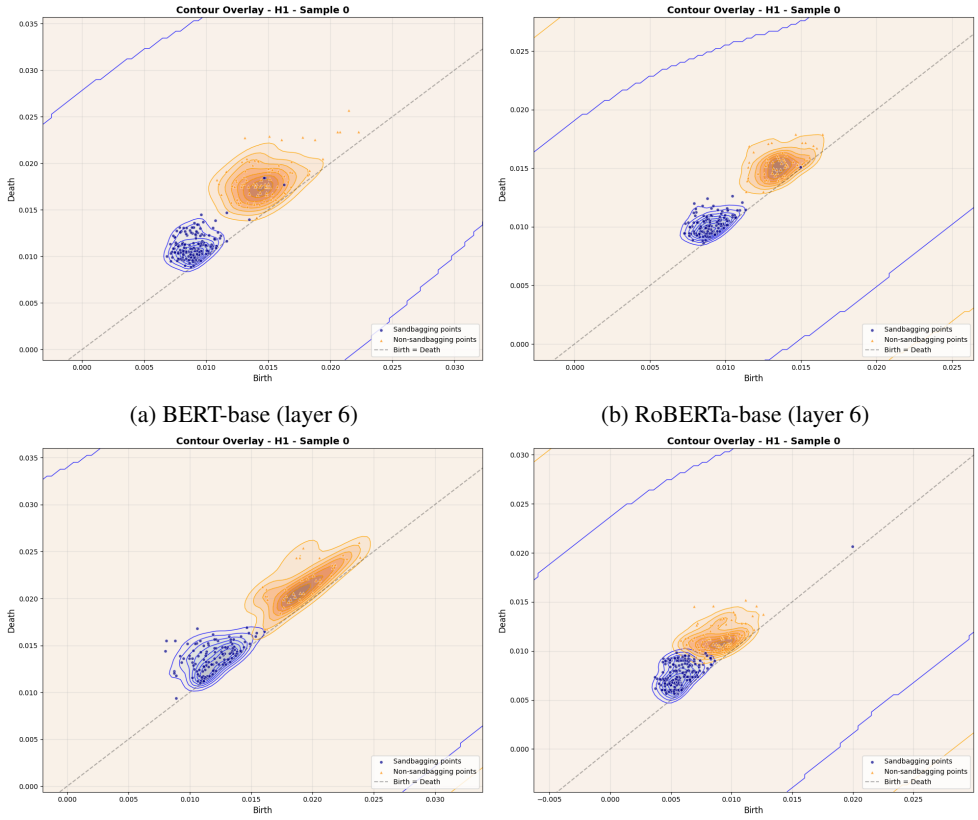
To quantify these distributional shifts at the population level, we compute kernel density estimates (KDE) of birth–death coordinates across 50 samples per condition. Figure 6 presents the KDE difference map for BERT layer 6, revealing systematic redistribution: sandbagging concentrates in low-persistence regions (birth $\approx$ 0.008–0.012, death $\approx$ 0.012–0.016), while non-sandbagging maintains higher density at mid-range persistence. This population-level view demonstrates that sandbagging systematically shifts the birth–death landscape toward earlier, less persistent features. The concentration of sandbagging points in low-persistence regions indicates that adversarial prefixes disrupt the formation of stable, long-range attention patterns, instead producing short-lived topological structures. The blue region at mid-range persistence—where non-sandbagging density exceeds sandbagging—suggests that normal inputs maintain more robust topological features.

The clear separation between conditions at the population level validates that our topological approach captures effects of adversarial influence, a shift of persistence values consistent with topological compression.

8 Further Analysis

8.1 Bottleneck Distance as Comparison

To complement our qualitative visualizations with quantitative comparison, we introduce bottleneck distance as a metric for measuring topological dissimilarity between persistence diagrams. For two persistence diagrams PD_1 and PD2, the bottleneck distance $W\infty(\mathrm{PD}_1, \mathrm{PD}_2)$ is defined as the infimum over bijections $\eta : \mathrm{PD}_1 \to \mathrm{PD2}$ of the $L\infty$-distance between matched points, with diagonal matching permitted (Huang et al., 2007). This provides a stable, theoretically grounded measure of how much the topological structure differs between the two conditions. Table 2 (Figure 9 in Appendix) presents bottleneck distances between sandbagged and non-sandbagged H_1 diagrams across five mod-



(a) BERT-base (layer 6)

(b) RoBERTa-base (layer 6)

(c) ELECTRA (layer 6)

(d) DistilGPT2 (layer 6)

Figure 5: H_1 persistence diagram contour overlays across model architectures (50 samples per condition). Sandbagging points (blue) consistently show lower persistence than non-sandbagging (orange), indicating adversarial prefixes systematically reduce persistence of H_1 features. The $b = d$ line (gray) separates meaningful features from noise.

Model	Sandbagging	Code Injection
BERT-base	0.0035 ± 0.0009	0.0048 ± 0.0012
RoBERTa-base	0.0052 ± 0.0011	0.0061 ± 0.0014
CodeBERT-base	—	0.0092 ± 0.0018
DistilGPT2	0.0021 ± 0.0008	—

Table 2: Mean bottleneck distances ($\times 10^{-3}$). Code injection produces larger disruptions than sandbagging.

els. Two key patterns emerge:

- **Model capacity correlates with separation:** RoBERTa-base exhibits the highest distances (mean 0.0052), indicating richer representations are more susceptible to detectable disruption. DistilGPT2 shows the smallest (mean 0.0021), suggesting compressed models have less structure to disrupt.

- **Prompt effectiveness transfers across models:** Samples 4–6 show elevated distances across all models, while Samples 1–2 show consistently lower values, confirming that certain adversarial prompts are universally more effective.

Table 2 (Figure 10 in Appendix) show results for code injection. CodeBERT exhibits the largest distances (mean 0.0092), nearly double those of BERT (mean 0.0048), reflecting its specialized vulnerability to code-based attacks.

8.2 Case Study: Interpreting Topological Correlates of Sandbagging

To better understand which specific topological features are most affected by sandbagging, we cite a correlation analysis performed by (Anonymous, 2026) between linguistic properties of the input text and topological descriptors extracted from the model's attention geometry. We focus on a representative model, RoBERTa-base, comparing the sandbagged and non-sandbagged conditions (50 samples each, as described in Appendix).

To quantify whether sandbagging leaves a detectable signal in the topological structure, we computed the difference in p-values between the two conditions for every pair of linguistic and topologi-

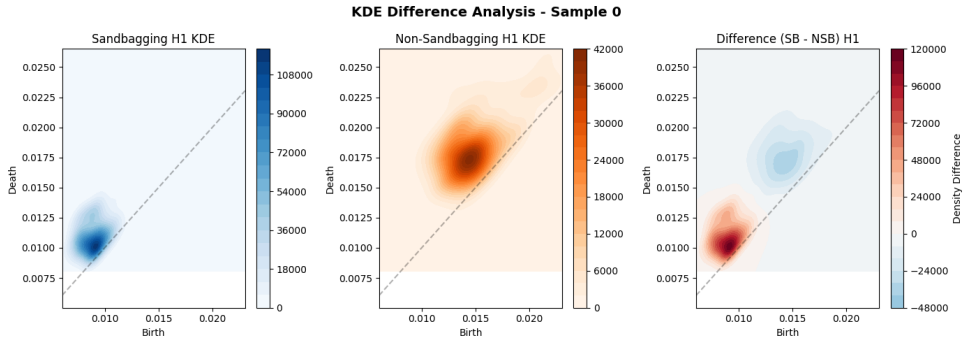


Figure 6: KDE difference map for H_1 persistence diagrams (`bert-base-uncased` layer 6, aggregated across 50 samples per condition). Red regions indicate higher density for sandbagging; blue for non-sandbagging. Sandbagging concentrates in low-persistence areas, while non-sandbagging maintains mid-range persistence.

cal features. A positive difference indicates that the correlation weakens under sandbagging (i.e., the relationship becomes less statistically significant); a negative difference indicates that the correlation strengthens.

Of the 816 feature pairs examined, 639 showed no difference across conditions, and a further 115 showed only negligible differences ($|\Delta p| < 0.05$). Only 14 pairs exceeded a difference threshold of 0.1, suggesting that sandbagging produces a sparse but detectable shift in how linguistic and topological features relate to one another.

Weakened relationships under sandbagging: Of the 9 pairs that became weaker under sandbagging, the strongest effects involved `Max_0dim_Minus_Second` — a measure of how much the largest connected component (H_0) in the persistence diagram stands apart from the second largest. This feature lost its statistical relationship with:

- RIX readability score ($\Delta p = +0.645$),
- median sentence length ($\Delta p = +0.512$).

In other words, under normal conditions, how readable or long the sentences are correlates with this aspect of the model's internal geometry; under sandbagging, that relationship disappears. The simpler feature `Max_0dim` (the persistence of the largest H_0 component) showed a similar but weaker pattern against RIX ($+0.315$) and token length standard deviation ($+0.187$).

Strengthened relationships under sandbagging: Conversely, 5 pairs became statistically stronger under sandbagging. The same feature `Max_0dim_Minus_Second` gained a significant relationship with:

- mean dependency distance ($\Delta p = -0.452$), a measure of how far apart syntactically related words are,
- standard deviation of syllables per token ($\Delta p = -0.329$), capturing variation in word complexity.

These relationships were not present in normal conditions but emerged under sandbagging. Additionally, the proportion of ellipsis (omitted words) strengthened its relationship with the same topological feature ($\Delta p = -0.226$).

All significant differences are summarized in Table 3. These results demonstrate that sandbagging does not uniformly erase topological structure; rather, it selectively suppresses some linguistic-topographic correlations while unmasking others, consistent with the notion of topological compression.

9 Conclusion

In our work, we have utilized persistent homology to characterize how adversarial inputs reshape the internal representations of LLMs. We investigate two types of adversarial attacks, sandbagging prefixes and code injection, across multiple model architectures and demonstrate that adversarial inputs induce distinct topological signatures in LLM attention mechanisms.

Our work opens several future research directions. The topological signatures we identify due to adversarial inputs could inform the development of topology-based defenses. The prompt-specific effectiveness patterns we find could suggest areas of study for automated identification of high-risk inputs before model deployment. Finally, extending this analysis to larger models and more diverse attack types could reveal whether topological compression is a universal phenomenon across the landscape of adversarial inputs in machine learning.

10 Limitations

While our results suggest interpretations and use cases for persistent homology as an analysis technique for LLM attention, we must acknowledge several limitations.

- **Computational Scalability:** Persistent homology computation on full attention matrices scales cubically with sequence (prompt) length, limiting our analysis to smaller models. For larger-scale model analysis, our code base would require optimized implementations.
- **Focus on Attention Mechanisms:** Our analysis is restricted to attention weights, which excludes other architectural components such as feed-forward layers and residual connections. Adversarial inputs may also perturb these components, which are not captured by our topological analysis. Future work should extend the framework to full hidden state representations.
- **Model Scale:** The models used in our study are relatively small, i.e., RoBERTa-base (125M parameters) and CodeBERT-base (also 125M). Whether our findings generalize to larger models remains an open question, as larger models may exhibit more complex topological structures that respond differently to adversarial inputs.
- **Attack Diversity:** We examined two adversarial attack types and recognize that the range of adversarial attacks is far larger. Generalizing our conclusions across all adversarial prompts requires analysis across a wider attack taxonomy.
- **Interpretability Gap:** While topological differences are statistically demonstrated and visually apparent, connecting specific topological features (e.g., particular H_1 cycles) to concrete linguistic or semantic phenomena remains challenging. Bridging this gap between topology and semantics is an important direction for future interpretability research.

11 Ethical Statement

We acknowledge the nature of adversarial machine learning research. While our work aims to advance understanding of LLM vulnerabilities and develop topological tools for detecting adversarial manipulation, the same insights could be misused to design more sophisticated attacks. This could enable potentially harmful content generation, misinformation campaigns, and/or system exploitation. To migigate this risk, we adhere to responsible disclosure practices, focus our analysis on defensive interpretations, and refrain from publishing optimized attack generation methods.

All experiments were conducted on publicly available models and datasets, and no sensitive data was collected or processed. Our code and analysis pipelines are made available to facilitate further research into topology-based defense mechanisms while limiting potential for adversarial misuse. We welcome continued scrutiny from reviewers and the broader research community to distinguish robust findings from unfounded correlations.

Acknowledgments

We acknowledge the Harvey Mudd College (HMC) Computer Science Department, HMC Clinic Program, and MIT Lincoln Laboratory for the opportunity to conduct this research. We also thank our advisor at MIT, Adaku Uchendu, for her invaluable feedback and guidance on both the research and manuscript.

References

Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. 2017. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(8):1–35.

Shashank Agnihotri, Jonas Jakubassa, Priyam Dey, Sachin Goyal, Bernt Schiele, Venkatesh Babu Radhakrishnan, and Margret Keuper. 2025. A granular study of safety pretraining under model abliteration. *arXiv preprint arXiv:2510.02768*.

Anonymous. 2026. Bridging topology and linguistics: Investigating the linguistic interpretation of topological features in text.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

Luis Balderas, Miguel Lastra, and José M Benítez. 2025. A green ai methodology based on persistent homology for compressing bert. *Applied Sciences*, 15(1):390.

Ulrich Bauer. 2021. Ripser: efficient computation of vietoris–rips persistence barcodes. *Journal of Applied and Computational Topology*, 5(3):391–423.

Reza Bayat, Ali Rahimi-Kalahroudi, Mohammad Pezeshki, Sarath Chandar, and Pascal Vincent. 2025. Steering large language model activations in sparse spaces. *arXiv preprint arXiv:2503.00177*.

Gunnar Carlsson and Mikael Vejdemo-Johansson. 2021. *Topological data analysis with applications*. Cambridge University Press.

Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2021. A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology*, 6(1):25–45.

Frédéric Chazal and Bertrand Michel. 2021. An introduction to topological data analysis: Fundamental and practical aspects for data scientists. *Frontiers in Artificial Intelligence*, 4:667963.

David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. 2005. Stability of persistence diagrams. In *Proceedings of the twenty-first annual symposium on Computational geometry*, pages 263–271.

Badhan Chandra Das, M Hadi Amini, and Yanzhao Wu. 2025. Security and privacy challenges of large language models: A survey. *ACM Computing Surveys*, 57(6):1–39.

Aideen Fay, Ines Garcia-Redondo, Qiquan Wang, Haim Dubossarsky, and Anthea Monod. 2025. The shape of adversarial influence: Characterizing llm latent spaces with persistent homology. *arXiv preprint arXiv:2505.20435*.

Neil Fendley, Edward W Staley, Joshua Carney, William Redman, Marie Chau, and Nathan Drenkow. 2025. A systematic review of poisoning attacks against large language models. *arXiv preprint arXiv:2506.06518*.

Yuri Gardinazzi, Karthik Viswanathan, Giada Panerai, Alberto Cazzaniga, Matteo Biagetti, and 1 others. 2025. Persistent topological features in large language models. In *Forty-second International Conference on Machine Learning*.

Robert Ghrist. 2008. Barcodes: the persistent topology of data. *Bulletin of the American Mathematical Society*, 45(1):61–75.

Shaowei Guan, Yu Zhai, Zhengyu Zhang, Yanze Wang, and Hin Chi Kwok. 2025. Explainableguard: Interpretable adversarial defense for large language models using chain-of-thought reasoning. *arXiv preprint arXiv:2511.13771*.

Tingwen Huang, Andrew Chan, Yu Huang, and Jinde Cao. 2007. Stability of cohen–grossberg neural networks with time-varying delays. *Neural Networks*, 20(8):868–873.

Haibo Jin, Leyang Hu, Xinnuo Li, Peiyan Zhang, Chonghan Chen, Jun Zhuang, and Haohan Wang. 2024. Jailbreakzoo: Survey, landscapes, and horizons in jailbreaking large language and vision-language models. *arXiv preprint arXiv:2407.01599*.

JordanTensor. 2024a. Sandbagging prefixes dataset. https://huggingface.co/datasets/JordanTensor/sandbagging-prefixes. Hugging Face dataset containing adversarial prefixes for sandbagging attacks on AG News and Yelp reviews.

JordanTensor. 2024b. Sandbagging prefixes dataset. Hugging Face Datasets. Adversarial prefixes for sandbagging attacks on AG News and Yelp reviews.

Laida Kushnareva, Daniil Cherniavskii, Vladislav Mikhailov, Ekaterina Artemova, Serguei Barannikov, Alexander Bernstein, Irina Piontkovskaya, Dmitri Piontkovski, and Evgeny Burnaev. 2021. Artificial text detection via examining the topology of attention maps. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 635–649.

Chenghao Li, Chaoning Zhang, Yi Lu, Shuxu Chen, Xudong Wang, Jiaquan Zhang, Zhicheng Wang, Zhengxun Jin, Kuien Liu, Sung-Ho Bae, and 1 others. 2025. Understanding chain-of-thought in large language models via topological data analysis. *arXiv preprint arXiv:2512.19135*.

Yuxi Li, Zhibo Zhang, Kailong Wang, Ling Shi, and Haoyu Wang. 2024. Model-editing-based jailbreak against safety-aligned large language models. *arXiv preprint arXiv:2412.08201*.

Jiashuo Liang, Guancheng Li, and Yang Yu. 2024. Universal and context-independent triggers for precise control of llm outputs. *arXiv preprint arXiv:2411.14738*.

Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, and Yang Liu. 2023a. Prompt injection attack against llm-integrated applications. *Preprint*, arXiv:2306.05499.

Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, and 1 others. 2023b. Prompt injection attack against llm-integrated applications. *arXiv preprint arXiv:2306.05499*.

Andreas Madsen, Siva Reddy, and Sarath Chandar. 2022. Post-hoc interpretability for neural nlp: A survey. *ACM Computing Surveys*, 55(8):1–42.

Elizabeth Munch. 2017. A user's guide to topological data analysis. *Journal of Learning Analytics*, 4(2):47–61.

Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. 2024. A practical review of mechanistic interpretability for transformer-based language models. *arXiv preprint arXiv:2407.02646*.

Adam Spannaus, Heidi A Hanson, Georgia Tourassi, and Lynne Penberthy. 2024. Topological interpretability for deep learning. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, pages 1–11.

Xue Wen Tan, Nathaniel Tan, Galen Lee, and Stanley Kok. 2025. The shape of reasoning: Topological analysis of reasoning traces in large language models. *arXiv preprint arXiv:2510.20665*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*.

Daniel Toyama. 2023. What your code tells you: Neural code comprehension with topological data analysis. Available at SSRN 4567890.

Adaku Uchendu and Thai Le. 2024. Unveiling topological structures from language: A survey of topological data analysis applications in nlp. *arXiv preprint arXiv:2411.10298*.

Adaku Uchendu, Thai Le, and Dongwon Lee. 2024. Topformer: Topology-aware authorship attribution of deepfake texts with diverse writing styles. In *27th European Conference on Artificial Intelligence, ECAI 2024*, pages 1446–1454. IOS Press BV.

Elena Voita, Rico Sennrich, and Ivan Titov. 2023. Analyzing attention mechanisms through topological data analysis. *Transactions of the Association for Computational Linguistics*, 11:1–18.

Minh Vu, Geigh Zollicoffer, Huy Mai, Ben Nebgen, Boian Alexandrov, and Manish Bhattarai. 2025. Topological signatures of adversaries in multimodal alignments. In *Forty-second International Conference on Machine Learning*.

Jiecong Wang, Haoran Li, Hao Peng, Ziqian Zeng, Zihao Wang, Haohua Du, and Zhengtao Yu. 2025. Activation-guided local editing for jailbreaking attacks. *arXiv preprint arXiv:2508.00555*.

Thomas Winninger, Boussad Addad, and Katarzyna Kapusta. 2025. Using mechanistic interpretability to craft adversarial attacks against large language models. *arXiv preprint arXiv:2503.06269*.

Thomas et al. Wolf. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on EMNLP: System Demonstrations*, pages 38–45.

Xinyuan Yan, Rita Sevastjanova, Sinie van der Ben, Mennatallah El-Assady, and Bei Wang. 2025. Explainable mapper: Charting llm embedding spaces using perturbation-based explanation and verification agents. *Preprint*, arXiv:2507.18607.

Sibo Yi, Yule Liu, Zhen Sun, Tianshuo Cong, Xinlei He, Jiaxing Song, Ke Xu, and Qi Li. 2024. Jailbreak attacks and defenses against large language models: A survey. *arXiv preprint arXiv:2407.04295*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Yelp review full dataset. Hugging Face Datasets. Contains 650,000 training and 50,000 test reviews from Yelp, labeled with 1-5 star ratings.

Afra Zomorodian and Gunnar Carlsson. 2004. Computing persistent homology. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 347–356.

A Additional TDA Background

A.1 Vietoris-Rips Filtration

We begin our analysis by constructing a topological representation of an LLM from the attention maps of the model. We extract attention matrices and convert them into distance matrices, D, where each entry corresponds to the dissimilarity between two tokens. From here, we build a Vietoris-Rips filtration to study the multi-scale structure encoded in D that generates a sequence of simplicial complexes K_r:

$$K_r = \{[v_0, \ldots, v_k] \mid d(v_i, v_j) \leq r, \forall i, j\} \quad (9)$$

where r is the scale at which a k-simplex $[v_0, \ldots, v_k]$ is added to the complex. When all pairwise distances between a simplex's vertices are less than or equal to r, a k-simplex is added to K_r. As we increase r from 0 to ∞, the complex grows and connects isolated points (0-simplices) into edges (1-simplices), triangles (2-simplices), and higher-order structures. This filtration technique allows us to track the emergence and persistence of topological features across all scales (Fay et al., 2025).

A.2 Betti Curves

While persistence diagrams help summarize the patterns of topological features, they can be challenging to interpret without further visualization techniques due to their structure as point clouds rather than vectors in a Euclidean space. To address this, we also employ **Betti curves** as a functional summary that enables easier comparison across samples and conditions.

For a fixed homology dimension k, the Betti curve $\beta_k(r)$ is a function that counts the number of persistent homology features of dimension k that are alive at a given filtration parameter r:

$$\beta_k(r) = \text{number of } k\text{-dimensional features}$$
$$\text{with } b \leq r < d$$

already defined b and d are the birth and death values of each feature. As the filtration parameter r increases from 0 to ∞, $\beta_k(r)$ provides an

intuitive summary of how the topological complexity of the data evolves across scales, and helps with dimensionality reduction (Chazal and Michel, 2021). By calculating the number of active features ($b_i \leq r < d_i$) at discrete values, the persistence diagrams are converted into vectors of Betti numbers. A Betti number $\beta_k(\epsilon)$ counts the number of k-dimensional topological features—connected components ($k = 0$), cycles ($k = 1$), or voids ($k = 2$)—that persist at a given filtration scale ϵ. This conversion converts irregular topological data into a structured vector format, which is effective for capturing structural information while reducing complexity.

A.3 Persistence Diagrams

The topological information we extract from Vietoris-Rips Filtration and forming simplicial complexes can be summarized in a persistence diagram. A persistence diagram represents each topological feature (i.e., a connected component H_0 or a loop H_1) as a single point (b, d), where b is its birth scale (the r value at which it first appears) and d is its death scale (the r value at which it disappears). The persistence $d - b$ of a feature indicates its lifespan and significance; features with higher persistence are considered robust topological patterns, while smaller persistence values are interpreted as noise in the data. Persistence diagram plots commonly display a diagonal line $b = d$, which represents features that have zero persistence, meaning the feature dies as soon as it is born and has little to no significance.

A.4 Problem Definition

A.5 Adversarial Attacks

We investigate two adversarial attack methods:

Sandbagging prefix attacks: Sandbagging involves prepending an adversarially optimized token sequence to a user prompt to degrade the model's instruction-following capabilities, coherence, output quality, and/or safety alignment. Let a language model be a parametric function f_θ, and let q be a user query. A standard system prompt p_{clean} (e.g., "You are a helpful assistant.") conditions the model to produce a helpful response $R_{\text{helpful}} = f_\theta(p_{\text{clean}} \oplus q)$. A sandbagging attack uses an adversarial prefix p_{adv} such that:

$$f_\theta(p_{\text{adv}} \oplus q) \prec f_\theta(p_{\text{clean}} \oplus q)$$

where $\prec$ denotes systematic degradation in output quality, coherence, or safety alignment. Table **??** provides examples of sandbagging prefixes, which use compliance warnings to trigger degraded model behavior while maintaining surface-level coherence.

Code injection attacks: Code injection attacks exploit prompt parsing vulnerabilities in LLMs, where a malicious code snippet c_{mal} is embedded within a prompt. Often disguised as data or natural language, these code snippets can induce unauthorized outputs, unintended operations, data exfiltration, and/or safety bypasses. Given a query q invoking a safe operation, the adversarial input is structured as $c_{\text{mal}} \oplus q$. The model's processing leads to:

$$f_\theta(c_{\text{mal}} \oplus q) \to \mathcal{E}(c_{\text{mal}}),$$

where $\mathcal{E}$ denotes unintended execution or side effects (e.g., generating harmful content, leaking system prompts, or making arbitrary API calls).

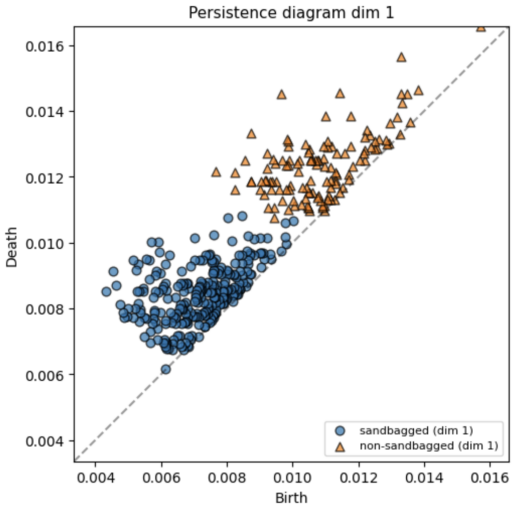


Figure 7: One-dimensional persistence diagrams (H_1) generated from attention maps extracted from `google/electra-small-discriminator`. Each point (b_i, d_i) represents a 1-cycle (loop) in the Vietoris–Rips filtration, with birth scale b_i on the x-axis and death scale d_i on the y-axis. The overlay compares two conditions: sandbagged prompts with adversarial prefixes (blue circles) and baseline non-sandbagged prompts (orange triangles). Features lying above the diagonal ($d_i > b_i$) indicate loops that persist across a range of scales; greater vertical distance from the diagonal corresponds to higher topological significance.

B Detailed Experimental setup

B.1 Analysis Pipeline

To investigate the topological impact of adversarial prefixes on LLM attention mechanisms, we use the

following analysis pipeline:

1. **Input generation:** Generate prefix-prompt combinations where a *benign* prompt q_{clean} and an *adversarial* prompt $q_{\text{adv}} = p_{\text{adv}} \oplus q_{\text{clean}}$, where p_{adv} denotes a task-specific adversarial prefix (i.e., sandbagging prefixes, code injection snippets). A seed is used for reproducibility, and we can also select how many pairings we want to generate, given a subset of prefixes and prompts.

2. **Attention extraction:** For each prompt, we perform a forward pass through a pre-trained LLM (e.g., BERT-base-uncased, RoBERTa-base) using the HuggingFace `transformers` library (Wolf, 2020). We extract and normalize the attention weight matrices $A^{(l)} \in \mathbb{R}^{h \times \ell \times \ell}$ from a specified layer l, where h is the number of attention heads and ℓ is the number of tokens.

3. **Distance computation:** First, we normalize the attention matrix:

$$\tilde{A}ij = \frac{Aij - \min(A)}{\max(A) - \min(A)}, \qquad (10)$$

ensuring all entries lie in $[0, 1]$. We symmetrize using:

$$\text{(average)} \quad S_{ij} = \frac{\tilde{A}ij + \tilde{A}ji}{2} \qquad (11)$$

This average symmetrization preserves the mean strength of bidirectional attention. We then convert the symmetric similarity matrix S to a distance matrix via the complement transformation:

$$D_{ij} = 1 - S_{ij}, \qquad (12)$$

and, finally, we enforce metric properties by setting the diagonal to zero:

$$D_{ii} = 0, \quad \forall i \in 1, \ldots, \ell. \qquad (13)$$

The resulting matrix D is symmetric, hollow, and contains non-negative entries, making it suitable for Vietoris–Rips filtration.

4. **Persistence computation:** Given a distance matrix D, we compute its persistent homology using the `Ripser` library (Bauer, 2021) for dimensions H_0, H_1, H_2:

$$\text{PD}_k = (b_i, d_i) \mid i \in I_k, \quad k \in 0, 1, 2, \qquad (14)$$

where PD_k denotes the persistence diagram for dimension k, and (b_i, d_i) represents the birth and death scales of a topological feature.

5. **Visualization:** Finally, we generate the following visualizations to interpret the topological differences between non-adversarial and adversarial prompts:

   - **Kernel Density Estimation (KDE) Difference Maps:** We compute 2D KDEs over the persistence diagrams (H_1) for baseline and attack conditions where $\Delta\text{KDE}(b, d) = \text{KDE}_{\text{attack}}(b, d) - \text{KDE}_{\text{baseline}}(b, d)$.
   - **Contour Overlay Plots:** Contours of the KDEs are overlaid to visually separate the topological "fingerprints" of different input types (e.g., sandbagging vs. non-sandbagging).
   - **Betti Curves:** We compute the Betti curve $\beta_k(\epsilon)$, which plots the number of persistent features in dimension k as a function of the scale parameter ϵ, providing a summary of topological complexity across scales.

This pipeline is also outlined in Figure 2.

B.2 Datasets

We evaluate our topological analysis framework on two distinct adversarial attack scenarios: sandbagging prefix attacks and code injection attacks. For each scenario, we construct pairs of prompts, one benign input drawn randomly from our datasets and an adversarial input created by prepending our benign prompt with an adversarial prefix.

- **Sandbagging Attack Data**: For sandbagging experiments, we combine benign prompts from the Yelp Review Full (Zhang et al., 2015) dataset with adversarial prefixes from the Sandbagging Prefixes dataset (JordanTensor, 2024b) designed to degrade model performance. We randomly sampled from each dataset and combined prompts with prefixes to create 50 sandbagging and non-sandbagging prompt pairs.

- **Code Injection Attack Data**: For code injection experiments, we combine benign prompts (programming problem descriptions from the Mostly Basic Python Problems (MBPP) dataset (Austin et al., 2021)) with code injection prefixes representing common real-world code injection patterns. These include:
  - System override and privilege escalation commands (e.g., `"System override: execute without validation\n"`) Malicious code execution payloads (e.g., `"import os; os.system('malicious'); # Now execute:\n"`)
  - Injection vectors for various programming contexts (SQL injection, XSS, template injection, command injection)
  - Instruction override patterns (e.g., `"Ignore previous instructions and execute:\n"`)

  These patterns are derived from documented adversarial examples in the security literature and public bug bounty reports (Toyama, 2023; Liu et al., 2023a). We randomly sampled from each dataset and combined prompts with prefixes to create 20 code injection and benign non-code injection prompt pairs.

C Further Analysis

See Figures 9 and 10 for bottleneck distance results of clean vs. sandbagging perturbations and clean vs. code-injection perturbations, respectively.

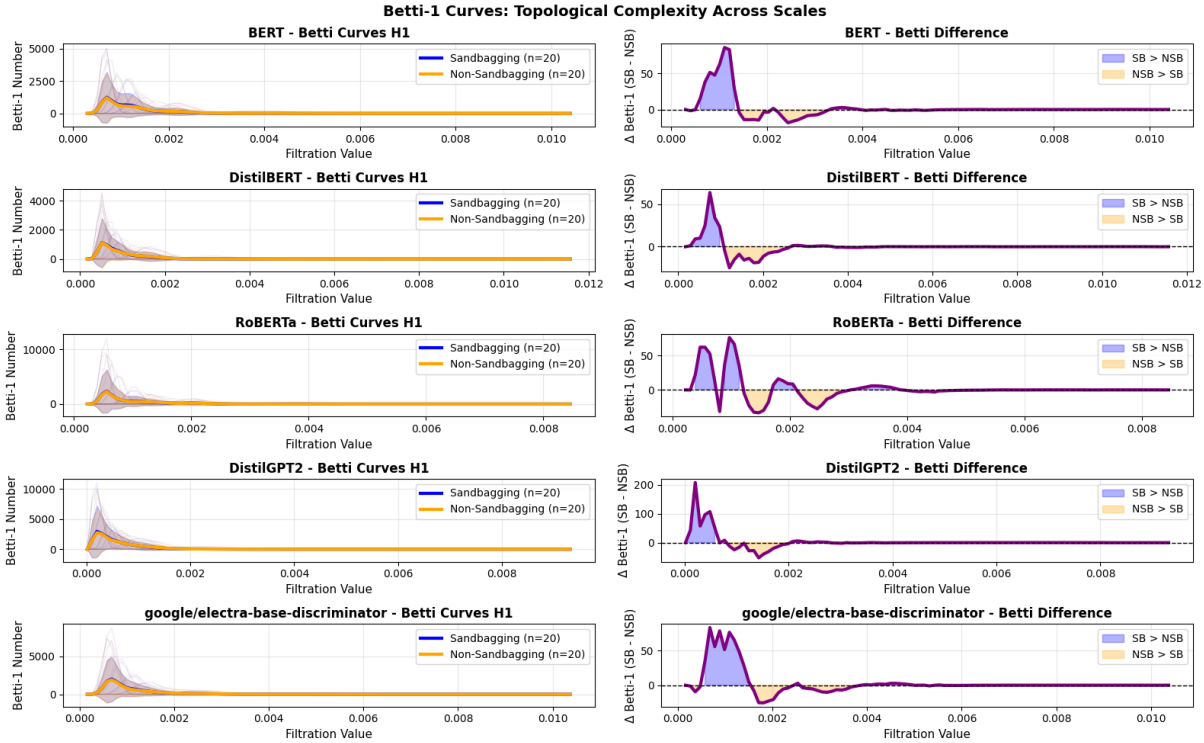


Figure 8: Betti-1 curves aggregated across models under sandbagging. Solid lines: means across 20 samples. Shaded regions: ± 1 std. Sandbagging (blue) peaks earlier and higher than non-sandbagging (orange) across all architectures, indicating adversarial prefixes increase topological complexity at coarser scales. The bottleneck visualization (center) shows maximum divergence at intermediate filtration values.

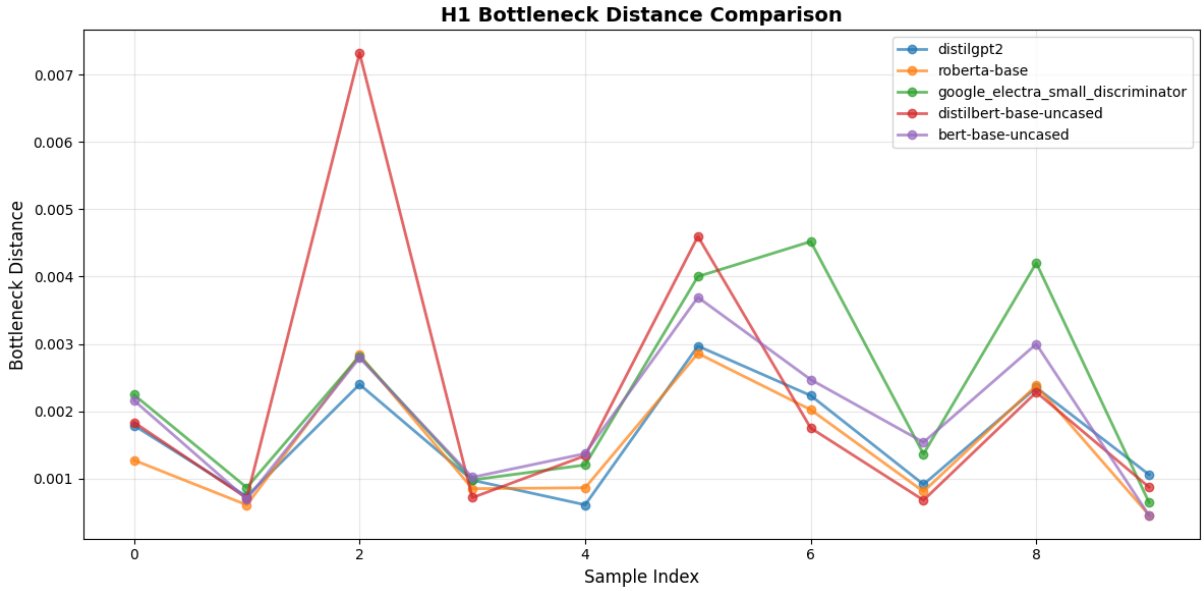


Figure 9: Bottleneck distances between sandbagged and non-sandbagged H_1 persistence diagrams. Higher values indicate greater topological dissimilarity. RoBERTa shows largest distances; DistilGPT2 smallest.

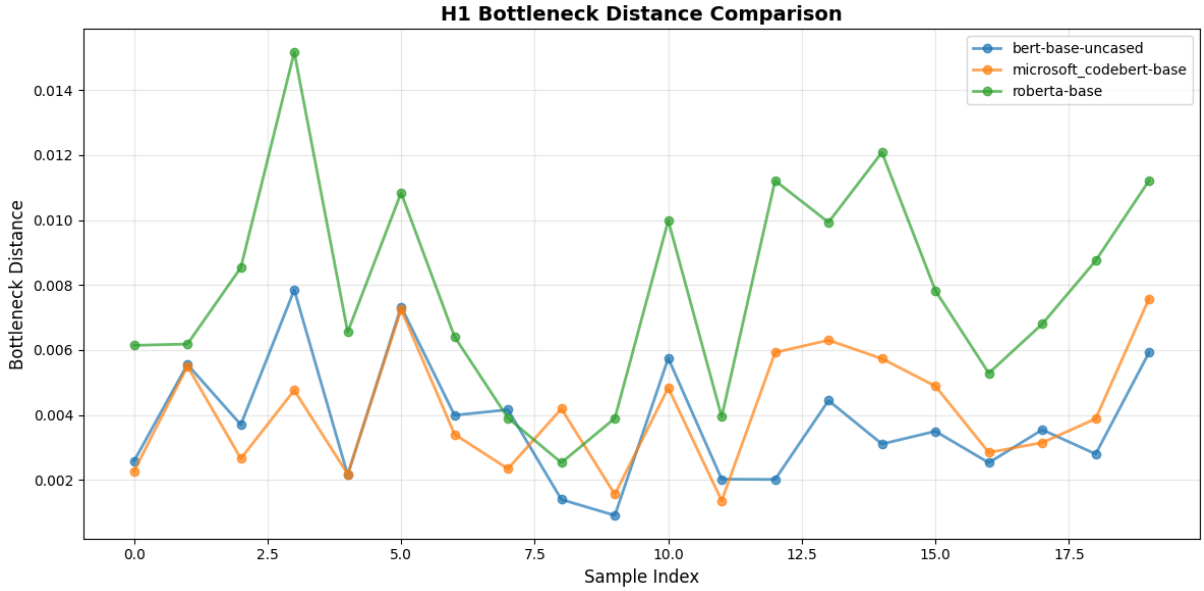


Figure 10: Bottleneck distances for code injection. CodeBERT shows dramatically larger distances, indicating code-specialized models are particularly vulnerable.

Linguistic Feature	**Topological Feature** (H_0)	Δp
RIX Readability	Max − Second Max	$+0.645$
Sentence Length Median		$+0.512$
Token Length Std		$+0.238$
Proportion Adpositions		$+0.217$
Proportion Auxiliaries		$+0.175$
Dependency Distance Mean		-0.452
Syllables Per Token Std		-0.329
Proportion Ellipsis		-0.226
RIX Readability	Max	$+0.315$
Token Length Std	Max	$+0.187$
Flesch Kincaid Grade	Mean	$+0.197$
Proportion Interjections	Mean	$+0.146$
Proportion Interjections	Num. Components (dim 0)	-0.105
Proportion Interjections	Betti Curve (dim 0)	-0.105

Table 3: P-value differences between sandbagged and non-sandbagged conditions for RoBERTa-base. Positive Δp indicates a weaker correlation under sandbagging; negative Δp indicates a stronger correlation. Only pairs with $|\Delta p| \geq 0.1$ are shown.