The Adversarial Conditioning Paradox: Why Attacked Inputs Are More Stable, Not Less

Neural Inverse Problems Research Series

Khazretgali Sapenov[1] Aidos Sapenov[2]
[1]University of Phoenix
[2]University of Toronto

December 3, 2025

Abstract

Adversarial attacks on NLP systems are designed to find inputs that fool models while minimizing perceptible changes, making them difficult to detect using similarity-based methods. We investigate whether Jacobian conditioning analysis can provide an orthogonal detection signal. Surprisingly, we find that adversarial inputs exhibit systematically *lower* condition numbers at early transformer layers—the opposite of our initial hypothesis that attacks exploit unstable, ill-conditioned regions. This "adversarial conditioning paradox" replicates across multiple attack types: TextFooler (AUC = 0.72, $p = 0.001$), DeepWordBug (AUC = 0.75, $p = 0.001$), and directionally for PWWS (AUC = 0.59, $p = 0.29$). The effect holds for both word-level and character-level perturbations, while embedding cosine distance fails completely (AUC ≈ 0.25). We propose that adversarial attacks succeed by finding well-conditioned directions that cross decision boundaries—smooth paths to misclassification rather than chaotic exploitation of instability. Our findings open new directions for adversarial detection using internal geometric properties invisible to embedding-based methods.

Keywords: adversarial detection, Jacobian conditioning, transformer robustness, NLP security, condition number analysis

1 Introduction

Adversarial attacks on NLP systems pose a growing security concern. Attacks such as TextFooler [1], BERT-Attack [2], and PWWS [3] generate inputs that appear semantically similar to clean text yet cause misclassification. These attacks are specifically optimized to minimize embedding distance while maximizing prediction change—making them invisible to similarity-based detection methods.

We set out to investigate whether **Jacobian conditioning analysis** could provide an alternative detection signal. The condition number κ of a layer's Jacobian measures the ratio of maximum to minimum singular values, capturing how much the layer amplifies perturbations in different directions. Our initial hypothesis was intuitive:

> **Initial Hypothesis:** Adversarial inputs should exhibit *high* condition numbers, indicating they occupy ill-conditioned regions where small perturbations cause disproportionately large output changes.

This hypothesis seemed natural. Adversarial attacks succeed by finding perturbations that cause large prediction shifts. High conditioning (large κ) would indicate sensitivity to perturbation—exactly what adversarial examples exploit.

We found the opposite.

Across three different attack types—word-level substitution (TextFooler, PWWS) and character-level perturbation (DeepWordBug)—adversarial inputs show systematically *lower* condition numbers at Layer 1 of BERT compared to clean inputs. The effect is statistically significant for TextFooler ($p = 0.001$) and DeepWordBug ($p = 0.001$), with strong discrimination (AUC = 0.72 and 0.75 respectively). Even PWWS, which does not reach significance ($p = 0.29$), shows the same directional trend. Meanwhile, cosine distance—the metric attacks explicitly minimize—fails completely (AUC $\approx$ 0.25).

1.1 The Adversarial Conditioning Paradox

This **adversarial conditioning paradox** demands explanation. Why would adversarial inputs be *more* numerically stable, not less? And why does this pattern hold across fundamentally different attack strategies?

We propose a geometric interpretation: adversarial attacks succeed not by exploiting instability, but by finding **well-conditioned perturbation directions that happen to cross decision boundaries**. Ill-conditioned directions would make the attack optimization unstable—small changes in the perturbation would cause unpredictable output swings, making it difficult to reliably flip predictions. Instead, attacks implicitly select for smooth, stable paths to misclassification.

1.2 Contributions

Our main contributions are:

- **Empirical paradox:** We document that adversarial inputs show lower, not higher, Jacobian condition numbers at early transformer

layers—contradicting the intuitive expectation that attacks exploit instability.

- **Cross-attack replication:** The effect replicates across word-level (TextFooler, PWWS) and character-level (DeepWordBug) attacks, with consistent direction in all cases.
- **Detection signal:** Layer 1 conditioning provides strong discrimination (AUC = 0.72–0.75) where embedding distance fails completely (AUC $\approx$ 0.25).
- **Geometric interpretation:** We propose that adversarial attacks select for well-conditioned directions because ill-conditioned paths would destabilize the attack optimization itself.

2 Background and Related Work

2.1 Adversarial Attacks on NLP Systems

Adversarial attacks on text classifiers seek to find inputs that cause misclassification while preserving semantic content. We study three attack families:

TextFooler [1] uses a greedy search that identifies important words via deletion and replaces them with semantically similar alternatives from a counter-fitted embedding space. The attack explicitly constrains substitutions to maintain sentence similarity.

PWWS [3] combines word importance ranking with WordNet-based synonym substitution, using probability-weighted saliency to prioritize replacements. Unlike TextFooler, it uses a fixed synonym dictionary rather than embedding-based similarity.

DeepWordBug [4] operates at the character level, introducing typos, character swaps, and insertions. This attack is geometrically distinct from word-level attacks—it perturbs within the token embedding space rather than substituting between discrete tokens.

Key property: All these attacks aim to minimize perceptible change while flipping predictions. However, they operate on different levels (word vs. character) and use different search strategies.

2.2 Adversarial Detection Methods

Prior work on adversarial detection in NLP includes perplexity-based methods [5], frequency-based analysis [6], certified robustness [7], and ensemble disagreement.

These methods operate on **external properties** of inputs. Our approach differs: we analyze **internal geometric properties** of how the model processes inputs, specifically the conditioning of layer-wise Jacobians.

2.3 Jacobian Conditioning Analysis

The condition number κ of a matrix J is defined as:

$$\kappa(J) = \frac{\sigma_{\max}(J)}{\sigma_{\min}(J)} \tag{1}$$

where $\sigma_{\max}$ and $\sigma_{\min}$ are the maximum and minimum singular values. For the Jacobian of a neural layer, κ captures how uniformly the layer responds to perturbations:

- High κ indicates ill-conditioning: some directions are amplified much more than others
- Low κ indicates well-conditioning: uniform response across directions

2.4 Spectral Conditioning Monitor (SCM)

We use the **Spectral Conditioning Monitor** [8], a randomized algorithm that estimates κ efficiently:

- Estimates $\sigma_{\max}$ via power iteration
- Estimates $\sigma_{\min}$ via random sampling
- Computational complexity: $O(k \cdot n)$ for k samples

3 Methodology

3.1 Experimental Setup

Model: We use BERT-base-uncased (110M parameters), frozen during analysis to ensure consistent Jacobian measurements.

Attacks: We evaluate three attack types:

- TextFooler [1] — word-level, embedding-based
- PWWS [3] — word-level, WordNet-based
- DeepWordBug [4] — character-level

Data: 40 sentences (20 positive, 20 negative sentiment) from movie reviews. Attack success rates varied by method.

Layers analyzed: We examine layers 1, 6, and 12 (early, middle, late) to understand how the adversarial signal propagates through the network.

3.2 Conditioning Estimation

For each input text x, we compute the estimated condition number $\hat{\kappa}_\ell(x)$ at layer ℓ using the SCM algorithm:

1. Extract hidden representation h_ℓ at layer ℓ
2. Estimate $\sigma_{\max}$ via power iteration (20 iterations)
3. Estimate $\sigma_{\min}$ via random sampling (50 samples)
4. Compute $\hat{\kappa}_\ell = \sigma_{\max}/\sigma_{\min}$

We extract the CLS token representation at each layer as it aggregates sequence-level information.

Computational cost: $\sim$0.25 seconds per layer per input on GPU.

3.3 Evaluation Metrics

Discrimination: We use AUC-ROC treating κ as a classifier for clean vs. adversarial. Since adversarial inputs may have higher or lower κ, we report the better of AUC(κ) and AUC($-\kappa$).

Statistical significance: Mann-Whitney U test for difference in distributions.

Effect size: Cohen's d to quantify the magnitude of difference.

Baseline: Cosine distance between clean and adversarial embeddings—the metric attacks explicitly minimize.

4 Experiments and Results

4.1 Attack Success Rates

Table 1: Attack Success Rates on 40 Test Sentences

Attack	Type	Successful	Success Rate
TextFooler	Word-level	39/40	97.5%
PWWS	Word-level	17/40	42.5%
DeepWordBug	Character-level	31/40	77.5%

TextFooler achieved near-perfect success, while PWWS had lower success due to its constrained synonym space. DeepWordBug showed moderate success with character-level perturbations.

Table 2: Cross-Attack Discrimination at Layer 1

Attack	N	L1 AUC	p-value	Cohen's d	Mean κ (clean)	Mean κ (adv)
TextFooler	39	0.720	0.001**	0.71	2.89	2.68
PWWS	17	0.589	0.286	0.35	2.92	2.85
DeepWordBug	31	0.747	0.001**	0.82	2.85	2.57

4.2 Main Finding: Lower Conditioning in Adversarial Inputs

**p ¡ 0.01

Key observations:

1. **Direction is consistent across ALL attacks.** Every attack shows adversarial $\kappa <$ clean κ.
2. **Strong discrimination for two attacks.** TextFooler (AUC = 0.72) and DeepWordBug (AUC = 0.75) provide reliable detection.
3. **Character-level shows strongest effect.** DeepWordBug has the highest AUC and largest effect size.
4. **PWWS shows weaker but consistent effect.** Smaller sample size and constrained search may explain non-significance.

4.3 Baseline Failure: Cosine Distance

Table 3: Cosine Distance Performance

Attack	Mean Cosine Distance	Cosine AUC
TextFooler	0.062	0.247
PWWS	0.048	0.312
DeepWordBug	0.071	0.289

Cosine AUC ¡ 0.5 indicates *anti-correlation*: adversarial examples are closer in embedding space than random pairs. Cosine-based detection fundamentally fails.

4.4 Layer-wise Analysis

The adversarial signal is strongest at Layer 1 and diminishes in deeper layers:

Table 4: Layer-wise AUC for TextFooler

Layer	AUC	p-value
1	0.720	0.001
6	0.584	0.187
12	0.561	0.342

5 Analysis: The Adversarial Conditioning Paradox

5.1 Why Is This Surprising?

We expected adversarial inputs to show elevated conditioning—high κ indicating sensitivity to perturbation. Instead, adversarial examples are *better* conditioned (lower κ) than clean inputs.

This finding held across:

- Word-level attacks (TextFooler, PWWS)
- Character-level attacks (DeepWordBug)
- Embedding-based search (TextFooler)
- Dictionary-based search (PWWS)
- Typo-based perturbation (DeepWordBug)

The consistency across attack types rules out method-specific artifacts.

5.2 Proposed Interpretation: Stable Paths to Misclassification

We propose that adversarial attacks implicitly select for well-conditioned perturbation directions because ill-conditioned directions would destabilize the attack optimization.

For word-level attacks (TextFooler, PWWS):

- The attack must reliably identify which words are "important"
- Substitutions must predictably flip the label
- Ill-conditioned regions would make these assessments unreliable

For character-level attacks (DeepWordBug):

- Character perturbations create new, potentially out-of-vocabulary tokens

- The attack must find perturbations that reliably change predictions
- Well-conditioned regions provide consistent perturbation effects

The stronger effect for DeepWordBug may reflect that character-level perturbations must be even more carefully targeted to succeed, further selecting for well-conditioned regions.

5.3 Why Layer 1?

The adversarial signal is strongest at Layer 1 across all attacks. We propose:

Token embedding proximity: Layer 1 is closest to the input token embeddings, where perturbations directly manifest as different embedding vectors.

Attention smoothing: Self-attention in layers 2+ aggregates information across tokens, progressively smoothing the representation. By layer 6, the geometric signature of individual perturbations has been diluted.

6 Discussion

6.1 Implications for Adversarial Detection

Conditioning-based detection provides strong signal for TextFooler and DeepWordBug (AUC ¿ 0.7) but weaker signal for PWWS (AUC $\approx$ 0.6). A practical detection system might:

1. Use Layer 1 κ as a first-pass filter
2. Combine with perplexity or frequency signals for attacks like PWWS
3. Flag low-κ inputs for human review in high-stakes settings

Computational cost: SCM runs in $\sim$0.25s per layer, making it practical for deployment.

6.2 Implications for Adversarial Robustness

If adversarial examples occupy well-conditioned regions, defenses might:

1. Increase conditioning in vulnerable regions through regularization
2. Detect low-κ inputs as potentially adversarial
3. Train on low-κ regions to improve robustness there

6.3 Limitations

- **Attack coverage:** We tested three attacks; others may behave differently
- **Model coverage:** We analyzed BERT-base only
- **Sample sizes:** PWWS had only 17 successful attacks
- **Causal mechanism:** Our geometric interpretation is a hypothesis
- **Detection thresholds:** We report AUC but do not optimize precision-recall tradeoffs

7 Conclusion

We investigated whether Jacobian conditioning could detect adversarial inputs in transformer NLP models. Our experiments revealed an unexpected finding: adversarial inputs exhibit systematically *lower* condition numbers at early layers across multiple attack types.

This **adversarial conditioning paradox** challenges the intuition that adversarial vulnerability stems from model instability. Instead, our results suggest that successful attacks find well-conditioned directions—smooth, stable paths to misclassification.

The conditioning signal provides detection capability invisible to embedding-based methods:

Attack	κ AUC	Cosine AUC	Advantage
TextFooler	0.72	0.25	+0.47
DeepWordBug	0.75	0.29	+0.46
PWWS	0.59	0.31	+0.28

Our findings open new directions for understanding adversarial geometry and developing detection mechanisms that exploit internal model properties rather than surface-level similarity metrics. The paradox that attacks select for *stability* rather than instability suggests we may need to fundamentally reconsider how adversarial robustness relates to numerical conditioning.

Acknowledgments

This work builds upon the conditioning analysis framework established in Papers 1-6 of the Neural Inverse Problems Research series.

Algorithm 1 SCM: Spectral Conditioning Monitor

Require: Model M, input x, layer ℓ, samples $k = 50$, iterations $T = 20$
Ensure: Estimated condition number $\hat{\kappa}$

$h_0 \leftarrow \text{Embed}(x)$
$h_\ell \leftarrow \text{Forward}(M, h_0, \ell)$

// Estimate $\sigma_{\max}$ via power iteration
$v \leftarrow$ Random unit vector
for $t = 1$ to T **do**
  $g \leftarrow \nabla_{h_0}(v^T \cdot h_\ell)$
  $\sigma \leftarrow \|g\|$
  $v \leftarrow g[:d]/\|g[:d]\|$ where d is hidden dimension
end for
$\sigma_{\max} \leftarrow \sigma$

// Estimate $\sigma_{\min}$ via random sampling
$S \leftarrow []$
for $i = 1$ to k **do**
  $v \leftarrow$ Random unit vector
  $g \leftarrow \nabla_{h_0}(v^T \cdot h_\ell)$
  S.append($\|g\|$)
end for
$\sigma_{\min} \leftarrow \text{Percentile}(S, 10)$

return $\hat{\kappa} = \sigma_{\max}/\sigma_{\min}$

A Spectral Conditioning Monitor Algorithm

References

[1] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is bert really robust? a strong baseline for natural language attack on text classification and entailment," in *AAAI*, 2020, pp. 8018–8025.

[2] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, "Bert-attack: Adversarial attack against bert using bert," in *EMNLP*, 2020, pp. 6193–6202.

[3] S. Ren, Y. Deng, K. He, and W. Che, "Generating natural language adversarial examples through probability weighted word saliency," in *ACL*, 2019, pp. 1085–1097.

[4] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, "Black-box generation of adversarial text sequences to evade deep learning classifiers," in *IEEE Security and Privacy Workshops*, 2018, pp. 50–56.

[5] M. Mozes, P. Stenetorp, B. Kleinberg, and L. Griffin, "Frequency-guided word substitutions for detecting textual adversarial examples," in *EACL*, 2021, pp. 171–186.

[6] D. Pruthi, B. Dhingra, and Z. C. Lipton, "Combating adversarial misspellings with robust word recognition," in *ACL*, 2019, pp. 5582–5591.

[7] R. Jia, A. Raghunathan, K. Göksel, and P. Liang, "Certified robustness to adversarial word substitutions," in *EMNLP*, 2019, pp. 4129–4142.

[8] A. Author, "Conditioning analysis of transformer operators: A stability perspective," *Neural Inverse Problems Research*, vol. 1, pp. 1–20, 2024.