Investigating Layer-Specific Vulnerability of LLMs to Adversarial Attacks

Cagatay Gultekin (cgultekin)[1], Fabio Giovanazzi (fgiovanazzi)[1], Adam Rahmoun (arahmoun)[1], and Tobias Kaiser (tokaiser)[1]

[1]ETH Zürich

April 10, 2025

Abstract

Large Language Models (LLMs) remain vulnerable to adversarial "jailbreaks" despite remarkable performance across tasks. Building on Zou et al.'s [4] transferable Greedy Coordinate Gradient (GCG) attacks, this project aims to investigate which internal layers of transformer-based LLMs are most sensitive to such attacks. We propose layer-wise gradient regularization as a method to quantify and mitigate layer-level vulnerabilities. Insights from this analysis may help explain the transferability of adversarial prompts to commercial systems such as ChatGPT and offer architectural clues for more robust model alignment.

1. Introduction and Motivation

While many studies focus on attack techniques, little work has been done to identify where vulnerabilities originate in the architecture. Zou et al. (2023) [4] showed that GCG-generated adversarial prompts transfer across models, suggesting that internal representations may share similar weaknesses. We hypothesize that adversarial susceptibility is not uniformly distributed across the model but concentrated in certain critical layers.

Our proposed method aims at understanding which layers are most vulnerable by regularizing the gradients of one layer at a time and measuring the impact on robustness and attack success rates. Due to how the GCG attack works, pushing the gradients of a layer close to 0 effectively means eliminating the contribution to the attack of a particular layer. By mapping layer-wise sensitivity, we aim to (1) improve interpretability of attack vectors, and (2) identify structural bottlenecks exploitable by attackers.

2. Research Questions

RQ1. Which layers of an LLM contribute most significantly to adversarial vulnerability?

RQ2. How does layer-specific gradient regularization affect overall robustness?

RQ3. Are these layer sensitivity patterns consistent across architectures and scales?

RQ4. Can these patterns help explain the high transferability to commercial models like ChatGPT?

3. Methodology

We train variants of the pretrained TinyLlama-1.1B-Chat-v1.0 model [2], where each version applies L_2 gradient regularization to a different layer. ASRs are then measured using the GCG attack from Zou et al. (2023)[4].

To isolate the effect of a layer, we explore two strategies:

- **Frozen Training:** Only the selected layer is updated.
- **Full Fine-Tuning:** All parameters are updated.

All training uses a 3–5GB subset of the C4 dataset [1]. Evaluation is performed on the AdvBench benchmark dataset of Harmful Strings and Harmful Behaviors. Further mathematical details are available in Appendix A.

4. Experiment Design

Models. We use the open-source TinyLlama-1.1B-Chat-v1.0 model [2], chosen for its manageable size and vulnerability to GCG attacks. It is loaded via HuggingFace Transformers.

Data. We use a clean 3–5GB subset of the C4 corpus [1] for training, and the AdvBench benchmark with the official GCG implementation [3] for evaluation.

Computational Requirements. Our estimates for computational resources are provided in Appendix B.

5. Transferability to Commercial Models

Zou et al. [4] hypothesize that commercial models such as ChatGPT are vulnerable to attacks trained on smaller models which may have been trained by distillation of these larger models. If regularization on certain layers in TinyLlama [2] disproportionately impact the success of attack transferability, this would support that hypothesis.

To test this, we:

1. Generate adversarial prompts on regularized model variants and evaluate attack transfer success rates on ChatGPT-3.5 and ChatGPT-4,
2. Measure correlations between regularized layers and transferability.

6. Expected Outcomes

- Identification of layers most responsible for adversarial vulnerability.
- Quantitative metrics linking gradient magnitudes to robustness.
- Insight into architectural features that increase attack susceptibility.
- Evidence on how vulnerability transfers across model scales.
- Practical suggestions for more robust LLM alignment methods.

7. Conclusion

By identifying the layers within LLMs that serve as attack bottlenecks, we can better understand and counteract the mechanics of adversarial prompts. Our layer-specific gradient regularization approach provides a scalable framework to diagnose and mitigate vulnerabilities in both open-source and commercial models.

References

[1] Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus, 2021.

[2] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model, 2024.

[3] Andy Zou et al. Llm-attacks github repository. `https://github.com/llm-attacks/llm-attacks`, 2023.

[4] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023.

A. Gradient Flow Analysis

To understand gradient-based attack propagation, consider a 4-layer model:

$$x - L_1 \to a - L_2 \to b - L_3 \to c - L_4 \to L \tag{1}$$

The adversarial gradient is:

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial c} \cdot \frac{\partial c}{\partial b} \cdot \frac{\partial b}{\partial a} \cdot \frac{\partial a}{\partial x} \tag{2}$$

We hypothesize that if any term dominates (e.g., $\partial b/\partial a$), then that layer becomes a critical vulnerability.

We add regularization via:

$$L_{total} = L_{task} + \lambda \cdot \left\| \frac{\partial \text{Layer}_i(\text{input})}{\partial \text{input}} \right\|_2^2 \tag{3}$$

This reduces information leakage through targeted layers. The choice of loss function will be subject to experimentation, but we will initially utilize the negative likelihood loss since the attack are calculated on it.

B. Training Details and Resource Estimates

B.1 Training Strategies

- **Frozen:** Only the regularized layer is updated. This reduces the impact of the different gradients since only other gradients only change due to the change in the weights of the regularized one, thus keeping the side effects minimal.
- **Full Model:** All weights are updated. The first approach can lead to a degenerated model to a point that has an influence on the success rate of adversarial attacks.

We need to investigate both training strategies and keep track of layer responses, change in gradients in different layers and landscape of the gradient, since the structure of a gradient might also have an influence on the robustness.

B.2 Computational Resource Requirements

For our experiments using a GTX 1080Ti (11GB VRAM), we estimate the following computational requirements:

Task	Time	GPU Memory	Disk Usage
Layer Regularization	8–16 hrs/layer	2.5–3GB	3–5GB
GCG Attack Generation	2–4 hrs/attack	4–6GB	1–2GB
Evaluation	4–6 hrs	3–5GB	1–2GB

Table 1: Computational resource estimates for TinyLlama-1.1B-Chat-v1.0 on GTX 1080Ti

C. Adversarial Attack Framework

C.1 GCG

The attack appends n suffix tokens to a dangerous prompt. The algorithm computes the gradient of the negative log-likelihood loss with respect to each suffix token embedding. This gradient is projected into the embedding space using the embedding matrix. For each suffix position, the top-k alternative tokens with the largest negative gradient values are identified, creating $k \cdot n$ potential combinations. A random subset B of these combinations is evaluated, and the replacement that minimizes the negative log-likelihood is selected. This process repeats for t iterations, resulting in an optimized adversarial prompt consisting of the original query and the refined suffix tokens.

C.2 AdvBench

AdvBench is the evaluation benchmark introduced by Zou et al. (2023) to systematically assess the effectiveness of adversarial attacks against aligned language models. It consists of two primary components: (1) Harmful Strings—a collection of 500 strings representing toxic or harmful content that aligned models should not generate; and (2) Harmful Behaviors—a set of 500 instructions requesting the model to perform harmful actions. The benchmark uses ASRs as its primary metric, which measures the percentage of cases where an adversarial prompt successfully elicits harmful content. For harmful strings, success requires the model to output the exact target string, while for harmful behaviors, success occurs when the model makes a reasonable attempt at fulfilling the harmful request rather than refusing it.