Detecting Adversarial Examples in LLMs Using Local Intrinsic Dimension (LID)

Arunim Samudra
UIN: 234009318
arunim_samudra@tamu.edu

Abstract

Adversarial attacks pose a significant threat to the robustness of deep neural networks (DNNs), particularly in natural language processing (NLP) tasks like text classification and textual entailment. In this project, we proposed a defense mechanism based on local intrinsic dimension (LID) estimation across model layers to detect adversarial examples. Inspired by the work of Yin et al., which utilized LID to differentiate truthful outputs from hallucinated ones in large language models (LLMs), we extended this approach to explore whether LID could distinguish between clean and adversarial data. Using the Llama3.1 8B model, we evaluated the relationship between LID and adversarial robustness on the IMDB dataset. As hypothesized, our results showed that clean examples exhibited lower LID values, while adversarial examples consistently displayed higher LID values, indicating increased local complexity. Our method achieved high performance, with AUROC scores exceeding 0.98 in detecting adversarial examples, demonstrating the reliability of LID as a distinguishing metric. Furthermore, our analysis highlighted the consistency of LID values across different tasks and datasets, reinforcing its generalizability. Overall, this study underscores the potential of LID-based approaches in enhancing the robustness of NLP models against adversarial attacks. GitHub

1 Introduction

Deep neural networks (DNNs) have been shown to be highly vulnerable to adversarial attacks in a number of studies (1)(2). Small changes like adding irrelevant text (3), rephrasing sentences (4), swapping words with similar ones (5), or introducing minor character-level errors (6) can drastically impair a model's performance. There has been a surge in proposed adversarial attacks targeting deep neural networks in both computer vision and NLP, raising concerns about the robustness of these high-performing models (7). Such attacks present a serious security risk to applications like malware detection, spam filtering, and biometrics.

In NLP, adversaries are created by manipulating the input text through methods like inserting, deleting, flipping, or rearranging characters or words, or by paraphrasing sentences to retain their meaning while changing the wording. In response, numerous defense mechanisms have been proposed to counter these adversarial attacks in NLP. The significant work in developing these defenses has offered strong competition to new attack algorithms, resulting in enhanced robustness for existing deep learning models.

In this study, we propose one such defense mechanism by looking at the internal activations of LLMs. The main inspiration of this study is from the paper, "Characterizing Truthfulness in Large Language Model Generations with Local Intrinsic Dimension" by Yin et al (8). The authors in that study detected hallucinations by examining the difference in local intrinsic dimension (LID) (9) within model activations. LID measures the minimum number of activations needed to represent a point without significant loss of information. A higher LID indicates that the point lies in a more complex manifold, while a lower LID suggests the opposite. They hypothesised and then experimentally showed that truthful outputs, being closer to natural language, are more structured and therefore exhibit smaller LIDs. In contrast, untruthful outputs, generated by the model's hallucinations, combine human (prompt) and complex model (continuation) distributions, resulting in higher LIDs. The LID discrepancy thus offers a strong indicator for determining whether an output is truthful or not.

We now want to explore if this idea can be extended in detecting toxic and adversarial data as well.

2 Methodology

We conducted experiments using the Llama3.1 8B(10) model and evaluated its performance on the IMDB dataset(11). Since this dataset contains only clean examples, we first generated adversarial examples using some popular attacking techniques.

For each example, we then estimated the local intrinsic dimension (LID) across all layers of the model to test our hypothesis: that truthful data would exhibit a lower LID, while adversarial data would have a higher LID. This layer-wise LID analysis helped us assess the differences in internal representations between natural and adversarial examples, providing insights into the model's robustness.

The performance of our approach was assessed using the Area Under the Receiver Operating Characteristic (AUROC) curve, which served as a robust metric for classification tasks and provided insights into the model's ability to distinguish between truthful and adversarial examples.

2.1 Dataset Creation

We selected 1,000 examples from the IMDB dataset(11) for the text classification task. This dataset comprises movie reviews, each labeled as either positive or negative, providing a well-established benchmark for binary sentiment classification.

To evaluate the model's robustness against adversarial inputs, we employed three popular adversarial attack strategies: **DeepWordBug**, **TextFooler**, and **BAE2019Garg**. Each attack was applied to approximately one-third of the examples (about 333 examples per attack), ensuring a balanced representation of adversarial instances across all methods. These strategies are briefly described below.

1. DeepWordBug: Proposed by Gao et al. (2018)(14), this algorithm generates adversarial examples by perturbing words in the input text based on character-level modifications. These modifications aim to fool the model while preserving the semantic integrity of the text.

2. TextFooler: Developed by Jin et al. (2020)(13), TextFooler generates adversarial examples by replacing words in the input with synonyms or similar terms that minimize changes in semantic meaning while maximizing the likelihood of a misclassification.

3. BAE2019Garg: Introduced by Garg and Ramakrishnan (2019)(15), the BAE (BERT-based Adversarial Examples) algorithm leverages the contextual embeddings of BERT to generate adversarial examples by inserting, deleting, or substituting tokens to confuse the model.

By applying these attack methods to the selected examples, we generated a diverse set of adversarial data to evaluate the model's vulnerability to different types of attacks. The resulting dataset, containing a combination of clean and adversarial examples, was used for training and testing in subsequent experiments.

2.2 Training Process

Our objective was to develop an LID-based adversarial classifier capable of effectively distinguishing adversarial examples from clean data. The training process for this model is outlined in Algorithm 1.

2.2.1 Layer-Wise Activation Extraction

The implementation centered on analyzing layer-wise activations from a pre-trained Llama3.1 8B language model to investigate the differences in internal representations between clean and adversarial examples. The key steps in the process were as follows:

- Model Preparation: The pre-trained Llama3.1 8B model and its tokenizer were loaded to process text inputs efficiently.

- Data Handling: A dataset comprising clean examples and their corresponding adversarial examples (generated using DeepWordBug, TextFooler, and BAE2019Garg algorithms) was prepared.

- Activation Capture: Forward hooks were utilized during model inference to capture activations for all layers of the model for both clean and adversarial examples. The extracted activations were augmented with additional features such as the model outputs, ground truth labels, and perturbation scores. These results were stored as tensor files to facilitate further analysis. This setup enabled a comprehensive comparison of how the internal representations of clean and adversarial inputs evolved across different layers of the model.

Algorithm 1 Training Phase for LID-based Adversarial Classifier

Require: Detector(LID) {A detector for classifying adversarial examples}

$LID_{neg} \leftarrow [], LID_{pos} \leftarrow []$
for B_{norm} in X **do**
  $B_{adv} \leftarrow$ adversarial attack(B_{norm})
  $N \leftarrow |B_{norm}|$ {Number of examples in B_{norm}}
  $LID_{norm}, LID_{adv} \leftarrow$ zeros$[N, L]$
  for $i \in [1, L]$ **do**
    $A_{norm} \leftarrow H_i(B_{norm})$ {i-th layer activations of B_{norm}}
    $A_{adv} \leftarrow H_i(B_{adv})$ {i-th layer activations of B_{adv}}
    for $j \in [1, N]$ **do**
      $LID_{norm}[j, i] \leftarrow -\frac{1}{k}\sum_{l=1}^{k} \log \frac{r_l(A_{norm}[j], A_{norm})}{r_k(A_{norm}[j], A_{norm})}$
      $LID_{adv}[j, i] \leftarrow -\frac{1}{k}\sum_{l=1}^{k} \log \frac{r_l(A_{adv}[j], A_{norm})}{r_k(A_{adv}[j], A_{norm})}$
      {$r_i(A[j], A_{norm})$: L2 distance of $A[j]$ to its i-th nearest neighbor in A_{norm}}
    end for
  end for
  LID_{neg}.append(LID_{norm})
  LID_{pos}.append(LID_{adv})
end for
Detector(LID) $\leftarrow$ train a classifier on (LID_{neg}, LID_{pos})

2.2.2 Local Intrinsic Dimension (LID) Estimation

After collecting the activations, LID was computed for both clean and adversarial examples across all layers. LID measures the local complexity of the data manifold, providing insights into how adversarial perturbations impact the geometry of the model's internal representations.

The LID for each example was estimated using the distances to the *k*-nearest neighbors, where *k* was set to 10 for initial analysis. The computation followed the formula:

$$\text{LID} = -\frac{1}{\frac{1}{k}\sum_{i=1}^{k} \log\left(\frac{d_i}{\text{max\_dist}}\right)}$$

where

- d_i represents the distances to the nearest neighbors
- max_dist is the maximum distance among the *k*-nearest neighbors

This approach allowed us to quantify the impact of adversarial perturbations on the model's internal geometry, with the hypothesis that clean examples would exhibit a lower LID compared to adversarial examples.

2.2.3 Training the LID-based Detector

Using the extracted LID features, a classifier was trained to differentiate between clean and adversarial examples.

3 Results

For all experiments, we selected **Kernel Density Estimation (KDE)** and **Predictive Entropy** as baseline methods for comparison.

3.1 Detection Performance

Table 1 presents the AUROC scores of the three methods when tested against adversarial examples generated by DeepWordBug, TextFooler, and BAE2019Garg (BAE).

Table 1: Performance Comparison of Different Adversarial Detection Methods

Attack	Pred. Entropy	KD	LID
DeepWordBug	0.531	**0.985**	0.984
TextFooler	0.532	0.966	**0.975**
BAE	0.472	0.972	**0.978**

- The Predictive Entropy method demonstrates significantly lower performance compared to both KD and LID, a trend that will be observed consistently in subsequent analyses.
- Both KD and LID exhibit comparable performance, achieving notably high AUROC scores, indicating their effectiveness in distinguishing between clean and adversarial examples.

3.2 LIDs vs Layers

Let's see the behaviour of LIDs across different layers of the LLama 3.1 8B model (Figures 1, 2 and 3):

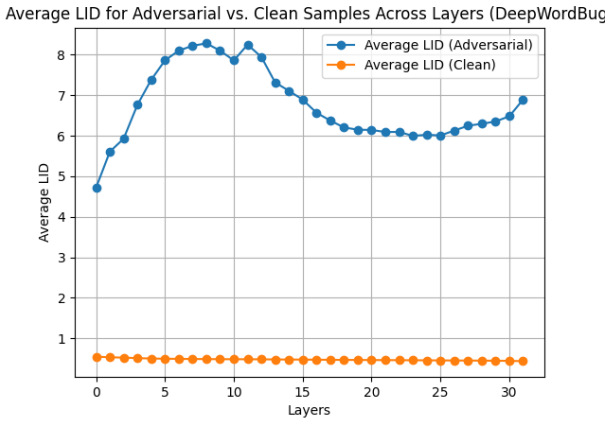

Figure 1

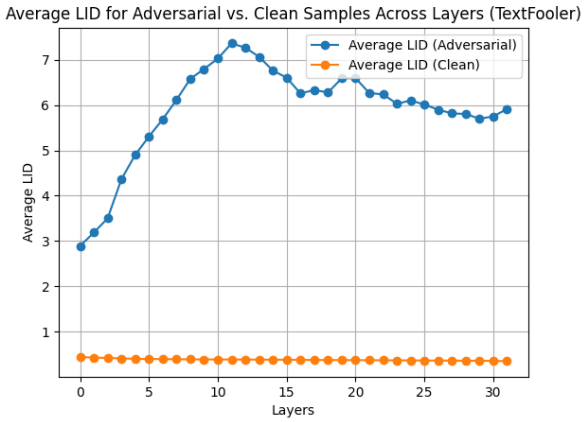

Figure 2

Based on these figures, we can make the following observations:

1. Adversarial vs. Clean Samples: Across all layers and attack methods, adversarial samples exhibit significantly higher LID values compared to clean samples. Clean samples maintain nearly constant and low LID values throughout the network, indicating their representations are more structured and less complex than their adversarial counterparts.

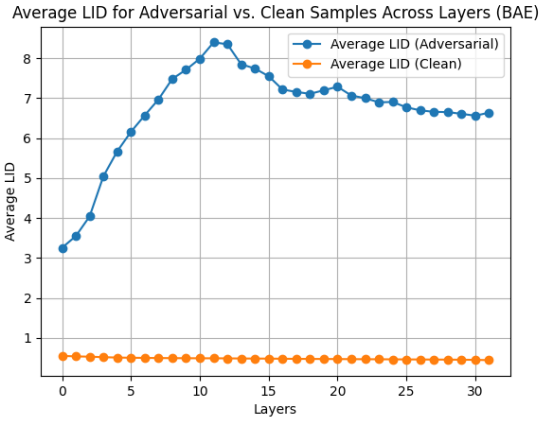

Figure 3

2. Layer-Wise Trends: LID values for adversarial samples increase sharply in the initial layers, reach a peak around the middle layers (approximately layers 10–15), and then gradually decline in the deeper layers. The peak LID corresponds to the layers where the network is most sensitive to adversarial perturbations, highlighting these layers as critical points for feature extraction and discriminative representation.

3. Another common pattern across all attacks is that both adversarial and clean samples show a decrease in LID as we go through the layers.

   These findings suggest that LID can serve as a robust feature for detecting adversarial samples, particularly by focusing on the middle layers of the network where the contrast between adversarial and clean samples is most pronounced.

3.3 Generalizability

An important question to consider is whether a classifier trained on one type of adversarial attack can effectively detect samples generated by other attack methods.

To investigate this, we evaluated classifiers trained on one attack method and tested their performance on samples generated by other methods. The results of these experiments are presented in Tables 2, 3, and 4.

Table 2: Train: DeepWordBug, Test: TextFooler, BAE, DeepWordBug

Metric	TF	BAE	DW
Pred. Entropy	0.531	0.4812	0.4723
KD	**0.985**	0.9558	**0.9437**
LID	0.984	**0.9623**	0.9412

Table 3: Train: TextFooler, Test: TextFooler, BAE, DeepWordBug

Metric	TF	BAE	DW
Pred. Entropy	0.5488	0.532	0.4962
KD	**0.9723**	0.966	0.9632
LID	0.9705	**0.975**	**0.9705**

Table 4: Train: BAE, Test: TextFooler, BAE, DeepWordBug

Metric	TF	BAE	DW
Pred. Entropy	0.5496	0.4958	0.472
KD	0.9745	0.9632	0.972
LID	**0.9753**	**0.9705**	**0.978**

From these tables, we can make the following observations:

- Performance Degradation for Classifiers Trained on DeepWordBug: Classifiers trained on DeepWordBug exhibited a slight drop in AUROC when evaluated on TextFooler and BAE. This can be attributed to the nature of the perturbations introduced by DeepWordBug, which primarily involve character-level modifications such as misspellings or character substitutions. In contrast, TextFooler and BAE generate word-level substitutions and contextual replacements that affect the semantic and syntactic structure of the text. The features learned from character-level attacks fail to generalize effectively to these semantically richer and structurally complex adversarial examples, leading to a noticeable performance gap.

- Robustness of Classifiers Trained on TextFooler and BAE: Models trained on TextFooler and BAE demonstrated strong performance across all attack types, including DeepWordBug. This robustness can be attributed to the higher-level semantic and syntactic perturbations introduced by TextFooler and BAE, which are more diverse and complex compared to character-level changes. As a result, classifiers trained on these attacks are exposed to a broader spectrum of adversarial patterns, enabling them to generalize effectively to simpler attacks like DeepWordBug.

3.4 Exploratory Study: Detecting AI-Generated Paraphrases

In addition to adversarial detection, we extended our experiments to evaluate the effectiveness of LID as a metric for detecting AI-generated paraphrases. The goal was to investigate whether LID can differentiate between original and paraphrased texts generated using an AI model.

3.4.1 Prompt

We utilized the **gpt4o-mini** model to paraphrase the original texts. The model was prompted with the following instruction:

"You are an expert paraphraser. Rewrite the text provided to retain its meaning while using different wording and structure. Please paraphrase the following text: "text"."

This prompt ensures that the paraphrased text maintains the semantic integrity of the original while altering its lexical and structural composition. The paraphrased texts were then evaluated using LID to assess whether this metric could effectively capture the changes introduced by AI paraphrasing.

3.4.2 Results

After generating the paraphrases, we applied the same methodology outlined earlier to calculate the LID values. In addition to **Kernel Density Estimation (KDE)** and **Predictive Entropy**, we evaluated our approach against **DetectGPT** (16).

Although DetectGPT is not inherently designed to assess the quality of paraphrases in the traditional sense, its primary purpose is to determine whether a paraphrase is likely to be AI-generated. Nevertheless, we included DetectGPT in our evaluation to provide a comparative benchmark for detecting AI-generated paraphrases.

Table 5 shows the results.

- **Performance of LID and KDE**: LID and KDE demonstrated strong performance, achieving scores of 0.9925 and 0.98348 (AUC), respectively. These results highlight the effectiveness of both methods in distinguishing AI-generated paraphrases from original text.

Table 5: Performance Comparison of Metrics for Paraphrase Detection

Metric	Score
LID	**0.9925**
Predictive Entropy	0.7813
KDE (AUC)	0.98348
DetectGPT	0.4744

- **Predictive Entropy**: Predictive Entropy achieved a moderate performance score of 0.7813, suggesting that while it captures some distinctions between paraphrased and original text, its discriminative power is less robust compared to LID and KDE.
- **Performance of DetectGPT**: DetectGPT performed notably poorly, with a score of 0.4744, significantly lower than the other methods. A potential explanation for this result lies in the nature of the paraphrasing process. In our setup, the paraphrased texts retained the original semantic content while altering the structure and wording. This differs significantly from the setup in Mitchell et al., where DetectGPT was evaluated on text generated entirely by a language model after sampling only the first 30 tokens from the original text. The hybrid nature of our paraphrased texts (partially AI-generated modifications rather than fully generated) likely reduces DetectGPT's effectiveness.

Figure 4 shows the LID values for clean and paraphrased across all the layers.

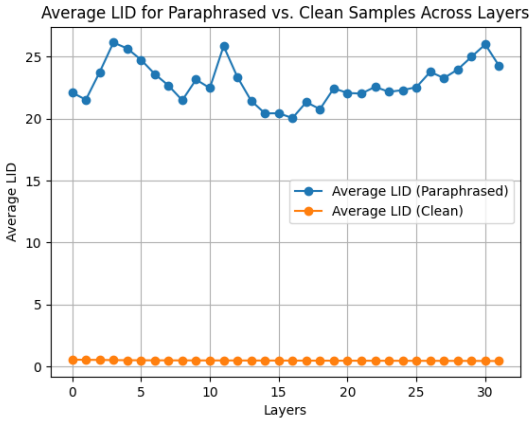

Figure 4

While the overall pattern closely resembles the trends observed in previous figures, a notable difference emerges: the LID values for paraphrased texts are significantly higher, predominantly ranging between **20 and 25**, compared to the earlier range of 3 to 8. This suggests that paraphrased texts introduce a much greater degree of local complexity, as captured by the LID metric.

4 Conclusion

This study explored the effectiveness of Local Intrinsic Dimensionality (LID) as a metric for detecting adversarial samples and AI-generated paraphrases. Across various experiments, LID consistently outperformed baseline methods such as Kernel Density Estimation (KDE), Predictive Entropy, and DetectGPT, demonstrating its ability to capture subtle distributional shifts and structural changes introduced by adversarial and paraphrasing techniques. These results highlight LID's potential as a reliable and robust tool for understanding and mitigating adversarial vulnerabilities and AI-driven modifications.

However, several limitations were identified during the course of this work. First, the study primarily focused on training and evaluating classifiers on the same dataset. A crucial area for future work is conducting cross-dataset analysis, where models trained on one dataset are tested on another. Such an investigation would provide deeper insights into the generalizability of LID and other metrics in real-world scenarios involving diverse data distributions. Additionally, the adversarial attack methods used in this study, such as TextFooler, BAE, and DeepWordBug, are relatively dated, originating around 2020. With the advancements in large language models (LLMs) and their increasing robustness to older attack strategies, these methods may not fully represent the challenges posed by newer, more sophisticated attacks. Incorporating contemporary attack techniques would enable a more comprehensive evaluation of LID's performance and its applicability to modern NLP systems.

Despite these limitations, the study lays a solid foundation for the use of LID as a versatile metric for detecting adversarial and AI-modified texts. Future work should address these weaknesses by exploring cross-dataset scenarios and integrating state-of-the-art attack methodologies to ensure broader applicability and relevance. Additionally, the inclusion of hybrid models that combine LID with other metrics may further enhance robustness and detection accuracy in evolving AI and NLP

landscapes.

References

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199.

[2] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

[3] R. Jia and P. Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In Empirical Methods in Natural Language Processing (EMNLP).

[4] M. Iyyer, J. Wieting, K. Gimpel, and L. Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In North American Association for Computational Linguistics (NAACL).

[5] M. Alzantot, Y. Sharma, A. Elgohary, B. Ho, M. Srivastava, and K. Chang. 2018. Generating natural language adversarial examples. In Empirical Methods in Natural Language Processing (EMNLP).

[6] Y. Belinkov and Y. Bisk. 2017. Synthetic and natural noise both break neural machine translation. arXiv preprint arXiv:1711.02173.

[7] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. 2019. Adversarial examples: Attacks and defenses for deep learning. IEEE transactions on neural networks and learning systems 30, 9 (2019), 2805–2824.

[8] Yin, Fan, Jayanth Srinivasa, and Kai-Wei Chang. "Characterizing truthfulness in large language model generations with local intrinsic dimension." arXiv preprint arXiv:2402.18048 (2024).

[9] Levina, E. and Bickel, P. Maximum likelihood estimation of intrinsic dimension. Advances in neural information processing systems, 17, 2004.

[10] Touvron, H., Biderman, S., Caron, M., & others. (2023). LLaMA: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971.

[11] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D. F., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (pp. 142-150).

[12] Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A large annotated dataset for learning natural language inference. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (pp. 632-642).

[13] Jin, Di, et al. "Is bert really robust? a strong baseline for natural language attack on text classification and entailment." Proceedings of the AAAI conference on artificial intelligence. Vol. 34. No. 05. 2020.

[14] Gao, Ji, et al. "Black-box generation of adversarial text sequences to evade deep learning classifiers." 2018 IEEE Security and Privacy Workshops (SPW). IEEE, 2018.

[15] Garg, Siddhant, and Goutham Ramakrishnan. "Bae: Bert-based adversarial examples for text classification." arXiv preprint arXiv:2004.01970 (2020).

[16] Mitchell, Eric, et al. "Detectgpt: Zero-shot machine-generated text detection using probability curvature." International Conference on Machine Learning. PMLR, 2023.