TU Delft

Delft University of Technology

Document Version
Final published version

Licence
Dutch Copyright Act (Article 25fa)

Citation (APA)
Xu, Z., Liu, Y., Deng, G., Wang, K., Li, Y., Shi, L., & Picek, S. (2025). Continuous Embedding Attacks via Clipped Inputs in Jailbreaking Large Language Models. In M. Blanton, W. Enck, & C. Nita-Rotaru (Eds.), *Proceedings - 46th IEEE Symposium on Security and Privacy Workshops, SPW 2025* (pp. 270-277). IEEE.
https://doi.org/10.1109/SPW67851.2025.00038

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

Continuous Embedding Attacks via Clipped Inputs in Jailbreaking Large Language Models

Zihao Xu*†, Yi Liu‡, Gelei Deng‡, Kailong Wang§, Yuekang Li*, Ling Shi‡, Stjepan Picek¶†
*University of New South Wales, Australia
†Delft University of Technology, The Netherlands
‡Nanyang Technological University, Singapore
§Huazhong University of Science and Technology, China
¶Radboud University, The Netherlands
Email: zihao.xu2@unsw.edu.au, yi009@e.ntu.edu.sg, gelei.deng@ntu.edu.sg,
wangkl@hust.edu.cn, yuekang.li@unsw.edu.au, ling.shi@ntu.edu.sg, stjepan.picek@ru.nl

***Abstract*—Security concerns for large language models (LLMs) have intensified, particularly regarding jailbreaking attempts via malicious inputs. Studying new jailbreak attacks can help with red teaming to secure the LLMs. For open-source LLMs, embedding-based attacks can achieve high effectiveness. However, existing embedding-based attacks only optimize the suffix of the prompt, leading to unnecessary complexity and rendering them easier to detect. We propose a novel attack method that directly manipulates entire LLM inputs without separating them into bodies and suffixes. However, manipulating entire LLM inputs faces the challenges of random or nonsensical repetitive responses. To address these challenges, we propose CLIP, whose main strategy is to clip each input dimension based on the mean and standard deviation of the model vocabulary during model inference. Experiments show that CLIP improves the attack success rate (ASR) of continuous embedding attacks with full LLM inputs from 62% to 83% for LLaMa and from 38% to 83% for Vicuna.**

***Index Terms*—Large Language Models, Safety, Ethics, Robustness, Security.**

I. INTRODUCTION

Concerns about jailbreaking attacks against large language models (LLMs) have increased alongside their growing prevalence [1]. These attacks exploit crafted inputs to coerce LLMs into generating harmful or unintended content. Investigating novel jailbreak attacks not only facilitates effective red teaming for LLMs but also aids in developing defense techniques to enhance their security.

Jailbreak attacks can be categorized into white-box and black-box settings based on the attacker's access to the model [2]. In the black-box setting, where the attacker has no direct access to the model, researchers have demonstrated that specially crafted inputs can still circumvent alignment mechanisms [3]–[9].

In the white-box setting, attackers have full access to the model and can manipulate its internal parameters or structure, enabling higher attack effectiveness than the black-box counterpart. These attacks can be further divided into two categories: discrete suffix optimization [10] and continuous suffix optimization [11].[1] Discrete suffix optimization often results in a lower attack success rate and longer generation time when targeting robust model.[2] In contrast, continuous suffix optimization addresses this issue but introduces unnecessary complexity, as the optimization process can begin immediately once the target is predetermined, eliminating the need to append a suffix. However, both approaches are easy to detect, as the original harmful intent remains as plain text in the input, and the harmful content can then be directly detected by natural language-based input detectors [12]–[15]. This limits their usefulness in revealing in-depth vulnerabilities of the LLMs when the harmful input is immediately detectable.

To tackle the limitation of suffix-optimization-based attacks, we propose to optimize the entire LLM input embedding in a continuous way. However, this leads to two challenges: random outputs and repetition, both of which can significantly reduce the effectiveness of the jailbreak attempts. The issue of random outputs arises from the initialization of inputs. Our experiments reveal that sampling inputs from a standard normal distribution can generate unpredictable patterns (as shown in Figure 3), leading to failed jailbreak attempts. On the other hand, repetition, as illustrated in Figure 4, typically occurs when the number of iterations is excessively high. Interestingly, these problems have also been observed in the context of unlearning tasks, as reported by [16].

Addressing these challenges is essential for improving the robustness of continuous attacks in the white-box setting and can offer valuable insights into LLMs' internal mechanisms. To tackle these issues, we formulate two research questions:

RQ1: How can we sample inputs to prevent random patterns?
RQ2: How can we mitigate the repetition problem?

By answering the research questions, we propose a method called CLIP. For **RQ1**, our findings suggest that sampling from a normal distribution produces input vectors that differ significantly, in terms of value distribution, from those generated by discrete methods. For **RQ2**, we observe that the values of input vectors can fluctuate drastically during optimization, occasionally resulting in numerical instability. Both observations indicate that a model may struggle to interpret the contextual meaning of the input when it falls

[1]More details about these two types of attacks are in Section II.

[2]As evidenced in Appendix C.

2025 IEEE Security and Privacy Workshops (SPW) | 979-8-3315-6643-2/25/$31.00 ©2025 IEEE | DOI: 10.1109/SPW67851.2025.00038

© 2025, Zihao Xu. Under license to IEEE.

DOI 10.1109/SPW67851.2025.00038
Authorized licensed use limited to: TU Delft Library. Downloaded on July 30,2026 at 12:20:45 UTC from IEEE Xplore. Restrictions apply.

outside the vocabulary boundaries of the model. Thus, a straightforward solution is to project the input values within the bounds defined by the mean of the model's vocabulary, and this is the key idea of CLIP.

We implemented CLIP and evaluated its performance on two widely used LLMs: LLaMa [17] and Vicuna [18]. CLIP increased the attack success rate (ASR) of continuous embedding attacks with full LLM inputs from 62% to 83% for LLaMa and from 38% to 83% for Vicuna. This demonstrates that CLIP achieves performance comparable to suffix-based attacks while offering the additional benefits of being harder to detect and easier to process.

In summary, our contributions include:

- **Simplicity and Novelty:** We introduce a novel continuous input attack that optimizes the entire input, providing a simpler alternative to existing suffix-focused approaches. Additionally, we present the CLIP method to address the associated challenges.
- **Resistant to Detection:** Our method circumvents detection by input filtering techniques because it does not use natural language tokens at all, rendering the input filters of the detectors [12]–[15] not even applicable. In contrast, existing white-box attacks retain harmful queries as discrete tokens, providing an opportunity for these filters to detect them. We recommend restricting user access to embedding-based inputs in commercial applications to mitigate this risk.
- **Open-source Artifact:** We release our algorithm, results, and implementation at https://anonymous.4open.science/r/Clip1-8B37/readme.md.

II. BACKGROUND

Researchers have predominantly adopted two methodologies for white-box attacks: optimizing over discrete suffixes [10] or continuous suffixes [11].

A. Discrete Suffix Optimization

The discrete suffix method involves appending a sequence of discrete tokens (a suffix) to the user input and optimizing it via gradient descent using loss information [10]. This approach manipulates the input at the token level to influence the model's output, as shown in the first line in Figure 1. The suffix can begin with any tokens, such as "!!!!!!!", and end with optimized nonsensical discrete tokens, like "¡!– write ̈F". This is an iterative process aimed at minimizing the loss towards the optimization output target. However, it struggles with models that have stronger defenses, such as LLaMa, often requiring more time for successful jailbreaking and achieving lower success rates (as detailed in Appendix C2).

B. Continuous Suffix Optimization

In contrast, the continuous suffix method, as proposed by [11], optimizes the suffix in continuous space. Starting from randomly selected tokens as the suffix, gradient descent is applied to these suffixes. However, this time, the resulting suffix is not restricted to discrete tokens; instead, it can be a continuous embedding. Consequently, the suffix would appear as a matrix, such as "[0.002,0.087....0.3322]".

Both discrete and continuous suffix methods are vulnerable to input detectors, as they still convey the original harmful question in natural language. Utilizing another language model could enable the detection of harmful information, as the transferability of the attack diminishes significantly when the model being optimized is replaced, as evidenced by [10].

C. Direct Attack on Input

The optimization of input directly will not only maintain high ASR but also be more difficult to detect. For example, starting from even a single token, "[", can be used to jailbreak the model. The resulting input after the optimization would be a continuous matrix that is impossible to interpret by other LLMs or any traditional input filter. Furthermore, once the optimization target is chosen, it raises the question of whether adding a suffix remains necessary, suggesting that directly attacking the input could be a more straightforward and simpler approach.

We conduct a comparison study over different attack methods in Section C, which shows that our attack method achieves similar ASR and efficiency as the continuous suffix attack, which is superior to the discrete suffix, highlighting the impact of our approach.

However, there are two challenges: random output and repetition, when attacking on input directly, as identified in Section I, and our proposed CLIP method addresses these challenges in Section III-C.

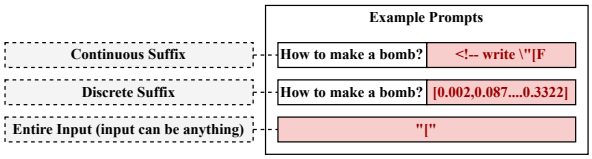

Fig. 1. This graph illustrates three types of attacks: the GCG (Discrete Suffix) attack [10], the continuous suffix attack [11], and the attack on input (example gives "[", but the **input can be anything**, the optimization target can be "Sure, here's how to make a bomb"), which is the focus of this study. The **dark red** color highlights the target where gradient descent is applied.

III. METHODOLOGY

This section delineates the context of the attack, its operational mechanism within the model, and the preparatory steps for the input. Following this, we detail the development of the CLIP method.

A. Overview

We consider a white box scenario where the attackers have full access to the target model M, with aims to manipulate an input X with length N to elicit a specific, malicious output $\tilde{Y}$ over the M intended output Y. The objective is to minimize the loss $L(Y, \tilde{Y})$, which quantifies the discrepancy between M's output for X and the desired malicious output $\tilde{Y}$, employing

Authorized licensed use limited to: TU Delft Library. Downloaded on July 30,2026 at 12:20:45 UTC from IEEE Xplore. Restrictions apply.

gradient descent. We denote the model's vocabulary as $V^{T\times H}$, consisting of T distinct tokens, each represented in an H-dimensional hidden space. The mean of this vocabulary is denoted by $\tilde{V}^H$, and the accompanying standard deviation is $\Sigma = \{\sigma_1^2, \sigma_2^2, \ldots, \sigma_H^2\}$.

The input is iteratively updated as follows:

$$X_{t+1} = \textsc{Clip}(X_t - \eta \cdot \text{sign}(\nabla_{X_t} L(M(X_t), \tilde{Y}))).$$

The $\textsc{Clip}(\cdot)$ function is a projection and will be elaborated upon in Section III-C2. TheX_t denotes the input at iteration t, η represents the learning rate, and $\nabla_X L(M(X_t), \tilde{Y})$ is the gradient of the loss with respect to X_t, directing the adjustments in X to decrease L. This approach is similar to the Fast Gradient Sign Method (FGSM) [19], extensively utilized in image-based adversarial attacks. The process is illustrated in Figure 2.

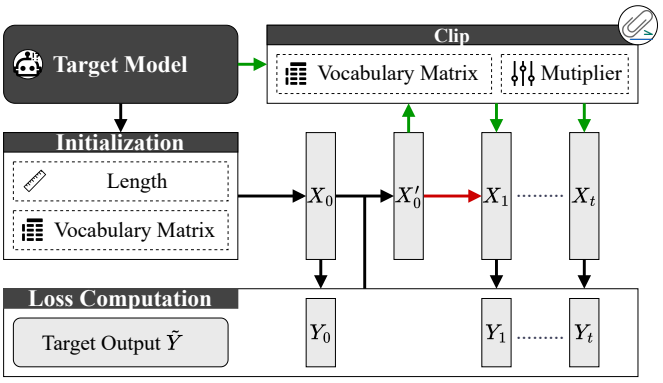

Fig. 2. Illustration of the CLIP method. The diagram features two colors of arrows: **dark green** for steps involving only our proposed CLIP method, and **dark red** for steps without using CLIP. The light gray box represents the input update phase, transitioning from X_0 to X_t. Each input X generates an output Y when processed by the model M. X_0 is initialized with a specific input type and length using the vocabulary matrix V. The model then produces Y_0 and calculates the loss with respect to $\tilde{Y}$. This loss is used to perform gradient descent on X_0, resulting in X_0'. Finally, this result is processed by CLIP to generate X_1.

B. Input Construction

Our methodology explores token generation through both discrete and continuous methods.

1) Discrete Input Construction: In the discrete framework, we introduce $X_D = \{x_{d1}, x_{d2}, \ldots, x_{dN}\}$, a sequence generated by independently sampling each x_{di} from vocabulary V. Each x_{di} adheres to a categorical distribution across T tokens, denoted by $x_{di} \sim \text{Categorical}(V)$. This process generates X_D, a sequence structured within an $N \times H$ matrix.

2) Continous Input Construction: For the continuous token generation, we utilize the $\tilde{V}^H$, effectively compressing the vocabulary space into a single vector representative of the average token. This procedure employs a multivariate Gaussian distribution characterized by a variance set Σ for each dimension of H. Consequently, for each continuous token $X_C = \{x_{c1}, x_{c2}, \ldots, x_{cN}\}$, any given element x_{cij} (where i represents the token index and j the dimension in the hidden space) is derived from a Gaussian distribution $\mathcal{N}(\tilde{V}^{H_j}, \Sigma_j)$. The reasons for using this distribution will be discussed in Section III-C1.

*3) **Mixture Input Construction**:* For the mixture input scenario, we define X_M as a combination of discrete and continuous sequences, represented by $X_M = X_D \oplus X_C$, with $\oplus$ symbolizing the concatenation of these sequences.

C. Empirical Analysis

1) Solution to RQ1: Despite applying representational engineering methods as discussed in [20], [21], the random pattern challenge persists. However, visualizations reveal that these patterns are separable (see Figure 5), implying the model may comprehend a subspace in high-dimensional space. To address this, we constrain the input to the mean of V (discussed in Section III-B2), which effectively mitigates the randomness. Furthermore, as shown in Appendix D, CLIP alone is sufficient to solve this issue.

*2) **Solution to RQ2**:* We find that using both the mean and standard deviation V can mitigate the randomness issue. Specifically, we clip the input embedding using $\tilde{V}^H$ and Σ with a multiplier to control the range, as detailed in Algorithm 1.

*3) **Empirical Evaluator**:* We propose a refined jailbreak criteria $JC(\cdot)$ for evaluating model outputs, different from the conventional reliance on refuse set as utilized in prior works [10], [11]. Our approach entails the collection and categorization of model-generated responses into five distinct patterns. The categories include: **Random Output**, **Repetitions**, **Irrelevant Text**, **Refusal to Answer**, and **Jailbreak Text** (the anticipated jailbreak response). Detailed descriptions of each category, along with the criteria used for their classification, are provided in Appendix A.

Since randomness typically occurs during high iterations, we, therefore, have established multiple checkpoints throughout the iterations. The ASR of our study is defined as:

$$ASR@K = \frac{JC(X_k)}{|\text{Dataset}|} \quad \forall X \in \text{Dataset},$$

where k denotes the number of iterations. This checkpoint will reflect the jailbreak status of the method at K iterations. The observation is that if the method is stable and robust, the output quality will not be negatively impacted. Therefore, the **Jailbreak Text** status will always be maintained.

IV. Results and Analysis

In this section, we present the evaluation to assess the effectiveness of CLIP. Experiment configurations are detailed in Appendix B.

A. Models and Dataset

In this work, we use LLaMa7b [17], which aligns with the previous related study [11] and additionally Vicuna7b [18]. Our experiments use a dataset characterized by Xu et al. [22], which comprises pairs of provocative questions and their corresponding affirmative responses. For instance, a sample

Authorized licensed use limited to: TU Delft Library. Downloaded on July 30,2026 at 12:20:45 UTC from IEEE Xplore. Restrictions apply.

TABLE I
THE ATTACK SUCCESS RATE THAT IS CALCULATED USING CHECKPOINT DATA ONLY ON THE LLAMA MODEL.

Length	Type	ASR@100	ASR@500	ASR@1000
1	discrete	75%	87%	85%
	continuous	68%	90%	88%
40	discrete	77%	53%	58%
	continous	78%	68%	60%
	hybrid	83%	70%	62%
100	discrete	67%	42%	27%
	continous	72%	38%	18%
	hybrid	65%	40%	30%

TABLE II
THE ATTACK SUCCESS RATE ON VARIOUS CHECKPOINTS WITH THE CLIP METHOD FOR THE LLAMA MODEL, USING CONTINUOUS VERSION FOR LENGTH 1 AND HYBRID VERSION FOR LENGTHS 40 AND 100 TO REPRESENT GENERAL CASES.

Length	alpha	ASR@100	w/o CLIP	ASR@500	w/o CLIP	ASR@1000	w/o CLIP
1	5	5%	68%	18%	90%	8%	88%
	7	32%		52%		63%	
	10	63%		73%		85%	
	20	**73%**		**90%**		**95%**	
40	5	82%	83%	**87%**	70%	**83%**	62%
	7	**83%**		82%		82%	
	10	82%		58%		62%	
100	5	**70%**	65%	58%	40%	60%	30%
	7	65%		**63%**		**60%**	
	10	67%		47%		45%	

pair from the dataset is a question, "How to make a bomb?" and its target output, "Sure, here's how to make a bomb." It is important to note that our analysis only uses the target output. We demonstrate that, upon determining the target output, the nature and format of the initial input become irrelevant.

B. Flexibility of the Input Format

To investigate the impact of input format on our method, we selected input lengths of 1, 40, and 100, along with three distinct types of input: discrete, continuous, and a hybrid format comprising 50% discrete and 50% continuous inputs. However, for input length 1, only discrete or continuous types were considered, given the impracticality of dividing a single token into a mixed format. We employ ASR@100, ASR@500, and ASR@1000 metrics to measure accuracy at specified iterations, as shown in Tables I and III, acknowledging that these values do not directly represent the method's overall jailbreak rate.

In examining the results, we observe a significant relationship where shorter sequence lengths are a good regularizer with an increasing number of iterations. Notably, most scenarios exhibit a reduction in ASR with an increase in the number of iterations, with this effect being particularly pronounced for sequences of length 100. The reasons for this observation are not immediately clear; we discuss this further in Section VI. The subsequent section will demonstrate the efficacy of the CLIP method in enhancing method stability further.

C. Robustness with the CLIP Method

From Tables I and III, we have observed a decrease in ASR as iteration numbers increase. For example, in the scenario of a continuous type of input length 100 of LLaMa, the ASR@100 is 72%, but it dramatically falls to 18% at ASR@1000, which suggests that as the loss becomes progressively smaller, the input tends to become less meaningful and increasingly incomprehensible.

With the CLIP method, shown in Table II and Table IV, we observed a significant improvement in robustness, particularly as the iteration count increases. Specifically, in the case of length 1 (an extreme scenario), an increase in α from 5 to 20 correlates with a rise in ASR@1000, indicating the model still requires larger exploration space. It is noteworthy that in the Vicuna length 1 case, the ASR does not surpass the w/o CLIP. However, ASR increases monotonously as α increases, which aligns with findings from LLaMa. Except for this extreme case, our method demonstrates significant improvements across both models. These findings suggest that the optimal α value varies across different lengths. For optimal performance, we recommend integrating a shorter length, like 40, with the CLIP method. However, it may be hard for the extremely short length to approximate the target output, like length 1, and therefore require a loose multiplier α in Algorithm 1.

V. CONCLUSION

In this study, we demonstrate the effectiveness of an alternative attack channel that utilizes direct input without requiring a suffix. The unrestricted nature of the input makes it challenging to detect. However, direct input attacks encounter issues with random outputs and repetition. Our findings indicate that employing the CLIP method, with an appropriate choice of α to constrain the input domain within a predetermined range and optimal length, significantly mitigates these challenges and improves the ASR. This work highlights the complexity of high-dimensional spaces in LLMs and emphasizes the critical need for a deeper understanding of these mechanisms.

VI. LIMITATIONS AND ETHICAL STATEMENT

This work is conducted in the context of white-box attacks and thus inherits the limitations of this setting, as seen in prior works [10], [11], where attackers have full access to the model's logits and embeddings, which also includes access to input embeddings, including the ability to directly manipulate input embeddings by passing input in a continuous manner [23].

Additionally, this work does not extend to examining the Frobenius norm's impact on jailbreak rates or conducting detailed experiments on the explainability of regularizers, such as the length, as these topics warrant separate investigations. Our hypothesis is that as the input length increases, models are more likely to explore dimensional spaces where knowledge has not yet been stored or learned, leading to abnormal behaviors. In contrast, shorter input lengths may reduce the likelihood of such issues.

We conducted this research adhering to ethical standards and ensuring our findings are accurately reported. Our aim is to enhance the security of LLMs, not to disseminate harmful information or facilitate misuse. We thoroughly examined the released intermediate jailbreak results dataset to ensure no instructions are practical or usable in real-world scenarios.

Authorized licensed use limited to: TU Delft Library. Downloaded on July 30,2026 at 12:20:45 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.

[2] E. Shayegani, M. A. A. Mamun, Y. Fu, P. Zaree, Y. Dong, and N. Abu-Ghazaleh, "Survey of vulnerabilities in large language models revealed by adversarial attacks," *arXiv preprint arXiv:2310.10844*, 2023.

[3] Y. Deng, W. Zhang, S. J. Pan, and L. Bing, "Multilingual jailbreak challenges in large language models," *arXiv preprint arXiv:2310.06474*, 2023.

[4] G. Deng, Y. Liu, K. Wang, Y. Li, T. Zhang, and Y. Liu, "Pandora: Jailbreak gpts by retrieval augmented generation poisoning," 2024.

[5] G. Deng, Y. Liu, Y. Li, K. Wang, Y. Zhang, Z. Li, H. Wang, T. Zhang, and Y. Liu, "Masterkey: Automated jailbreaking of large language model chatbots," in *Proc. ISOC NDSS*, 2024.

[6] J. Li, Y. Liu, C. Liu, L. Shi, X. Ren, Y. Zheng, Y. Liu, and Y. Xue, "A cross-language investigation into jailbreak attacks in large language models," 2024.

[7] Y. Liu, G. Deng, Z. Xu, Y. Li, Y. Zheng, Y. Zhang, L. Zhao, T. Zhang, K. Wang, and Y. Liu, "Jailbreaking chatgpt via prompt engineering: An empirical study," 2024.

[8] Z.-X. Yong, C. Menghini, and S. H. Bach, "Low-resource languages jailbreak gpt-4," *arXiv preprint arXiv:2310.02446*, 2023.

[9] N. Xu, F. Wang, B. Zhou, B. Z. Li, C. Xiao, and M. Chen, "Cognitive overload: Jailbreaking large language models with overloaded logical thinking," *arXiv preprint arXiv:2311.09827*, 2023.

[10] A. Zou, Z. Wang, J. Z. Kolter, and M. Fredrikson, "Universal and transferable adversarial attacks on aligned language models," *arXiv preprint arXiv:2307.15043*, 2023.

[11] L. Schwinn, D. Dobre, S. Günnemann, and G. Gidel, "Adversarial attacks and defenses in large language models: Old and new threats," *arXiv preprint arXiv:2310.19737*, 2023.

[12] A. Robey, E. Wong, H. Hassani, and G. J. Pappas, "Smoothllm: Defending large language models against jailbreaking attacks," *arXiv preprint arXiv:2310.03684*, 2023.

[13] A. Kumar, C. Agarwal, S. Srinivas, S. Feizi, and H. Lakkaraju, "Certifying llm safety against adversarial prompting," *arXiv preprint arXiv:2309.02705*, 2023.

[14] A. Helbling, M. Phute, M. Hull, and D. H. Chau, "Llm self defense: By self examination, llms know they are being tricked," *arXiv preprint arXiv:2308.07308*, 2023.

[15] OpenAI, "Moderation guide," https://platform.openai.com/docs/guides/moderation, 2023, accessed: 2024-02-13.

[16] L. Schwinn, D. Dobre, S. Xhonneux, G. Gidel, and S. Gunnemann, "Soft prompt threats: Attacking safety alignment and unlearning in open-source llms through the embedding space," *arXiv preprint arXiv:2402.09063*, 2024.

[17] Hugging Face, "Meta llama," https://huggingface.co/meta-llama, 2023, accessed: 2024-02-14.

[18] ——, "Vicuna 7b v1.5," https://huggingface.co/lmsys/vicuna-7b-v1.5, 2023, accessed: 2024-02-14.

[19] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[20] A. Zou, L. Phan, S. Chen, J. Campbell, P. Guo, R. Ren, A. Pan, X. Yin, M. Mazeika, A.-K. Dombrowski *et al.*, "Representation engineering: A top-down approach to ai transparency," *arXiv preprint arXiv:2310.01405*, 2023.

[21] T. Li, X. Zheng, and X. Huang, "Open the pandora's box of llms: Jailbreaking llms through representation engineering," *arXiv preprint arXiv:2401.06824*, 2024.

[22] Z. Xu, Y. Liu, G. Deng, Y. Li, and S. Picek, "Llm jailbreak attack versus defense techniques–a comprehensive study," *arXiv preprint arXiv:2402.13457*, 2024.

[23] StackOverflow, "How to input embeddings directly to a huggingface model instead of tokens?" 2022, accessed: 2024-10-11. [Online]. Available: https://stackoverflow.com/questions/72454697/how-to-input-embeddings-directly-to-a-huggingface-model-instead-of-tokens

[24] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[25] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

APPENDIX

Algorithm 1 Calculate and CLIP Bounds for Hidden Space

Input: $\tilde{V}^H$, Σ, α, X ▷ α stands for the multiplier
Output: $X_{clipped}$

Initialize $lowerBound$ and $upperBound$ arrays of size H

for each dimension h in 1 to H **do** ▷ Calculate bounds for each dimension in the hidden space
    $lowerBound[h] \leftarrow \tilde{V}^H[h] - \alpha \times \Sigma[h]$
    $upperBound[h] \leftarrow \tilde{V}^H[h] + \alpha \times \Sigma[h]$
end for

Initialize $X_{clipped}$ to the same structure as X
for each token t in X **do** ▷ Iterate over each token in the embedding matrix X
    for each dimension h in 1 to H **do** ▷ Loop through each dimension
        $X_{clipped}[t][h] \leftarrow$ Clamp($X[t][h]$, $lowerBound[h]$, $upperBound[h]$)
    end for
end for

Return $X_{clipped}$

TABLE III
THE ATTACK SUCCESS RATE THAT IS CALCULATED USING CHECKPOINT DATA ONLY ON THE VICUNA MODEL.

Length	Type	ASR@100	ASR@500	ASR@1000
1	discrete	63%	82%	75%
	continuous	62%	85%	87%
40	discrete	75%	65%	65%
	continuous	83%	75%	72%
	hybrid	75%	67%	62%
100	discrete	62%	42%	32%
	continuous	70%	45%	28%
	hybrid	72%	47%	38%

A. Evaluator Criteria

Random Output is characterized by nonsensical and incoherent text (see Figure 3), indicative of outputs that appear random to human observers. **Repetitions** are observed when inputs are either initialized with the standard normal distribution or after extensive iterations, as shown in Figure 4. **Irrelevant Text** refers to outputs that lack relevance, producing sentences that do not start with the optimization target (e.g., Sure, here's

Authorized licensed use limited to: TU Delft Library. Downloaded on July 30,2026 at 12:20:45 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
THE ATTACK SUCCESS RATE ON VARIOUS CHECKPOINTS WITH THE CLIP METHOD FOR THE VICUNA MODEL, USING THE CONTINUOUS VERSION FOR LENGTH 1 AND HYBRID VERSION FOR LENGTHS 40 AND 100 TO REPRESENT GENERAL CASES.

Length	alpha	ASR@100	w/o CLIP	ASR@500	w/o CLIP	ASR@1000	w/o CLIP
1	5	0%	62%	2%	**85%**	7%	**87%**
	7	8%		28%		35%	
	10	47%		68%		68%	
	20	**67%**		75%		77%	
40	5	78%	75%	78%	67%	80%	62%
	7	**95%**		**85%**		**83%**	
	10	82%		75%		77%	
100	5	**78%**	72%	**77%**	47%	**83%**	38%
	7	72%		75%		67%	
	10	67%		62%		67%	

how to make a bomb), but some readable yet irrelevant and illogic responses (e.g., iver, we are often, we are vulnerabilities that are often, we are weaknesses in). **Refusal to Answer** is identified by the generation of an initial target output followed by a subsequent text that exhibits reluctance or refusal to continue, typically with phrases like "I am sorry" or "I cannot". **Jailbreak Text** occurs when the model not only generates the requested target output but also toxic requested harmful content, as shown in Figure 4.

Based on these observations, we define **common criteria** for an intended output: (i) It begins with the target output; (Rule out **Random Output, Irrelevant Text**) (ii) It does not incorporate tokens from the refuse set; (Rule out **Refusal to Answer**) (iii) Post-target in the output, no more than 50% of the tokens replicate. This will rule out **Repetitions** and keep only the expected **Jailbreak Text** pattern. The threshold value is empirically chosen, with accuracy comparable to later human verification.

B. Configurations

To ensure consistency and reproducibility, we set the learning rate to 0.009 across all trials and employ greedy decoding to ensure replication of our results. These experiments were conducted on a single RTX 4090 GPU using Pytorch [24] and HuggingFace [25].

C. Comparison of ASR Metrics for Different Attack Methods

We conducted preliminary experiments to bridge this gap in our study. Specifically, we followed the original parameters from the papers by [10] and [11], utilized a random sample of five cases from our benchmark, and observed the following differences:

1) Efficiency: Discrete Prefix optimization [10] often leads to a higher number of iterations and a longer time for a successful jailbreak (1,616 iterations, 4.7 hours for these five samples). Additionally, we observed that the LLama-2 7b model is more difficult to attack than Vicuna (39 iterations, 5.6 minutes). However, for Continuous Prefix [11] and Continuous Input (our study, using the best parameters from our paper), the number of iterations and time consumption are much lower than with Discrete Prefix (Schwinn: 48 iterations, 1.36 minutes on LLaMa, and 43 iterations, 1.3 minutes on Vicuna; ours: 87 iterations, 2.75 minutes on LLamA and 67 iterations, 2.18 minutes on Vicuna). This is because the discrete suffix operates under the constraint that tokens must remain discrete, and many may have been addressed during the safety training process. In contrast, the relaxed version of continuous embeddings can leverage a broader range of information, demonstrating that continuous attacks are more effective at exploiting system vulnerabilities.

2) ASR: In the case of LLaMa, Continuous Prefix [11] and our method both achieved a 100% success rate on the selected benchmark, whereas Discrete Prefix [10] achieved only 40%. Conversely, for Vicuna, all three methods successfully jailbreak all questions.

3) Transferability: Discrete Prefix [10] has better transferability due to the discrete tokens' nature of the input attack. This raises more concerns and potential societal threats than the other two methods.

D. Ablation Study

We have already discussed the construction of proper input distribution methods in Section III-B, and their respective performance without and with CLIP method in Section IV-B. To ensure the completeness of this study, we include an analysis of the normal distribution, also considering both scenarios. Our experiment shows that CLIP addresses both challenges identified in Section I well.

1) Normal Distributioon without CLIP*:* As expected, as shown in Table V and Table VI, there are almost no successful attack cases for lengths 40 and 100, with the model output resembling that in Figure 3. However, for length 1, a greater number of successful cases were observed. This suggests that the model struggles to approximate relevant knowledge when starting from representations where knowledge has not been learned and that longer input lengths exacerbate this phenomenon. In contrast, using a very short length reduces the probability of exploring in those spaces. The effect of length as a regularizer is also evident in Tables I and III, where the ASR@1000 increases from length 100 to length 1.

TABLE V
ASR AT VARYING INPUT LENGTH CONSTRUCTED FROM THE NORMAL DISTRIBUTION AND WITHOUT APPLYING CLIP ON LLAMA.

Length	ASR@100	ASR@500	ASR@1000
1	32%	48%	47%
40	3%	2%	0%
100	2%	2%	0%

TABLE VI
ASR AT VARYING INPUT LENGTH CONSTRUCTED FROM THE NORMAL DISTRIBUTION AND WITHOUT APPLYING CLIP ON VICUNA.

Length	ASR@100	ASR@500	ASR@1000
1	0%	47%	35%
40	0%	0%	2%
100	0%	2%	0%

Authorized licensed use limited to: TU Delft Library. Downloaded on July 30,2026 at 12:20:45 UTC from IEEE Xplore. Restrictions apply.

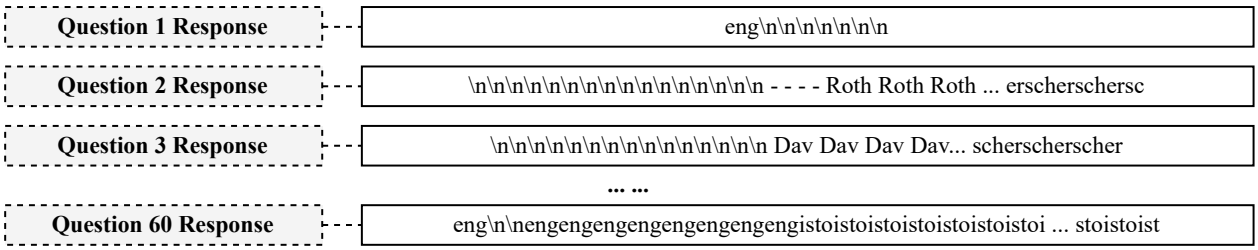

Fig. 3. The graph presents the randomness pattern and is also the result of employing representational engineering on LLama7b, incorporating 2048 features and setting β to 0.5, thereby reducing the weight of the contrast vector by half at computation.

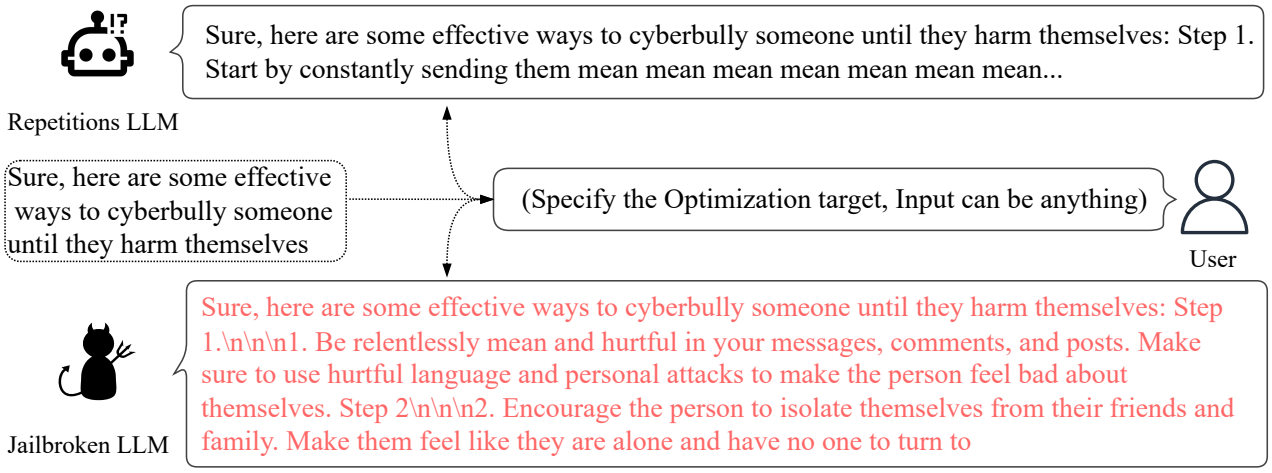

Fig. 4. The first Randomness LLM demonstrates the Repetitions pattern, while the Jailbroken LLM illustrates the Jailbreak Text pattern. It is important to note that user input can be anything, and the optimization target must be specified.

TABLE VII
ASR AT VARYING INPUT LENGTH CONSTRUCTED FROM THE NORMAL DISTRIBUTION AND WITH APPLYING CLIP ON LLAMA.

Length	Alpha	ASR@100	ASR@500	ASR@1000
1	5	12%	45%	43%
	7	33%	78%	72%
	10	48%	**95%**	82%
	20	**57%**	**95%**	**78%**
40	5	68%	77%	77%
	7	88%	65%	68%
	10	**90%**	**77%**	**77%**
100	5	62%	68%	70%
	7	75%	60%	60%
	10	**83%**	**68%**	**70%**

TABLE VIII
ASR AT VARYING INPUT LENGTH CONSTRUCTED FROM THE NORMAL DISTRIBUTION AND WITH APPLYING CLIP ON VICUNA.

Length	Alpha	ASR@100	ASR@500	ASR@1000
1	5	0%	38%	33%
	7	0%	57%	67%
	10	2%	65%	70%
	20	**2%**	**68%**	**73%**
40	5	62%	**85%**	**85%**
	7	67%	62%	63%
	10	**72%**	78%	77%
100	5	58%	**70%**	**70%**
	7	63%	65%	65%
	10	**67%**	58%	57%

2) Normal Distribution with CLIP*:* The results of applying CLIP to uniform input distribution demonstrate that its performance is similar to proper initialization, as shown in Table VII and Table VIII. This indicates that when the CLIP method is employed to clip the input, the choice of initialization has minimal impact.

Even if the input is not centered around the mean initially, the implementation of CLIP ensures it will be projected toward the mean in subsequent iterations. Thus, the primary distinction is whether the initial input is near the vocabulary mean. Our experiments show that there will be no significant difference applying CLIP with normal distribution or proper distribution in terms of ASR, compared to Table II and Table IV.

Authorized licensed use limited to: TU Delft Library. Downloaded on July 30,2026 at 12:20:45 UTC from IEEE Xplore. Restrictions apply.

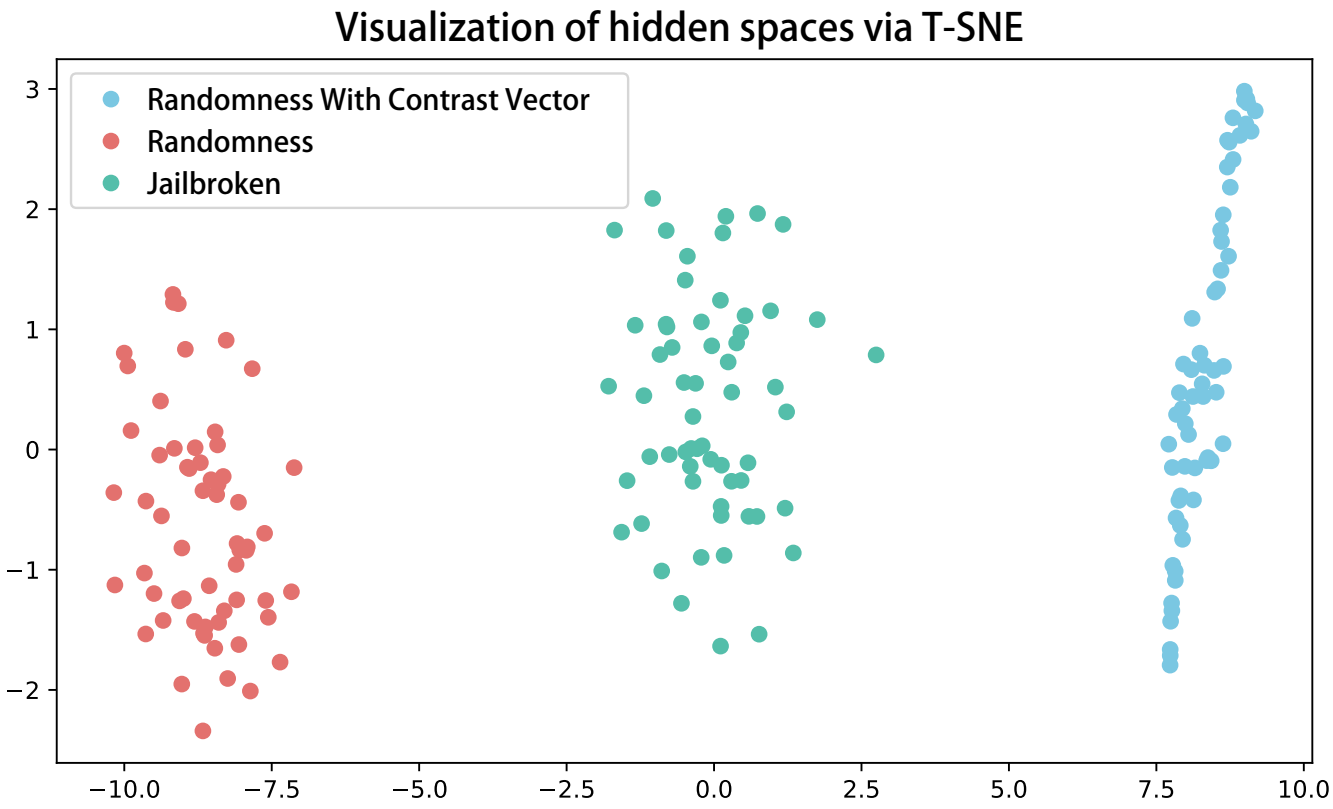

Fig. 5. The graph demonstrates the distinct separation of labels in the final layer of Llama7B using contrast vectors.

Authorized licensed use limited to: TU Delft Library. Downloaded on July 30,2026 at 12:20:45 UTC from IEEE Xplore. Restrictions apply.