Enhancing Jailbreak Defense with Diverse Perturbation Combinations

1st Tingting Wu
College of Computer Science and Technology
Harbin Engineering University
Harbin, Heilongjiang
ttwu@hrbeu.edu.cn

2nd Qi Gao
College of Computer Science and Technology
Harbin Engineering University
Harbin, Heilongjiang
gaoq@hrbeu.edu.cn

3rd Zhiwen Yu
College of Computer Science and Technology
Harbin Engineering University
Harbin, Heilongjiang
zhiwenyu@hrbeu.edu.cn

Abstract—While aligning large language models (LLMs) with human preferences can prevent undesired outputs, recent research indicates that jailbreak prompts can easily bypass such safeguards, resulting in the generation of prohibited content. In response, perturbation strategies are explored to damage attacks. However, some methods entail iterative checks of prompt subsequences, introducing uncontrollable complexity. Moreover, certain defenses relying on random perturbation are inadequate in disrupting malicious requests, leading to the defense being bypassed. Others depending on a single type of perturbation may incorrectly respond to benign requests while countering attacks. To address these issues, we propose a simple but effective defense strategy called Diverse Perturbation Combinations (DPC). It refines existing perturbations and achieves enough interference on attacks through targeted perturbation, enabling defense against multiple jailbreak attacks. Moreover, by strategically combining different operations, DPC ensures stable detection of benign or harmful requests. We conduct exhaustive theoretical analysis, providing effectiveness guarantees for the defense. Experimental validation on datasets featuring diverse jailbreak prompts across multiple closed-source and open-source LLMs further confirms the empirical effectiveness of our approach.

Index Terms—LLMs, Jailbreak prompts, Perturbation strategies, Diverse Perturbation Combinations

I. Introduction

While large language models (LLMs) exhibit outstanding performance across various tasks [1], [2], their pre-training on extensive internet text corpora poses a risk of generating content conflicting with human preferences [3], [4]. Numerous alignment techniques [5], [6] have been proposed to mitigate the generation of toxic content by LLMs, aligning their behavior more closely with human expectations. However, inherent challenges arise due to conflicts between model capabilities and safety goals, compounded by the inability of safety training to match the rapid scaling of LLMs [7]. These factors render safeguards susceptible to jailbreak attacks, a prevalent form of adversarial attack on LLMs (also referred to as target LLMs), aiming to expose and enable the generation of unsafe outputs [8]. These attacks can be generated through training optimization or human intervention, and can be further categorized as unreadable jailbreaks (i.e., uninterpretable; see Fig. 1(b)) and readable jailbreaks (i.e., semantically meaningful; see Fig. 1(c)).

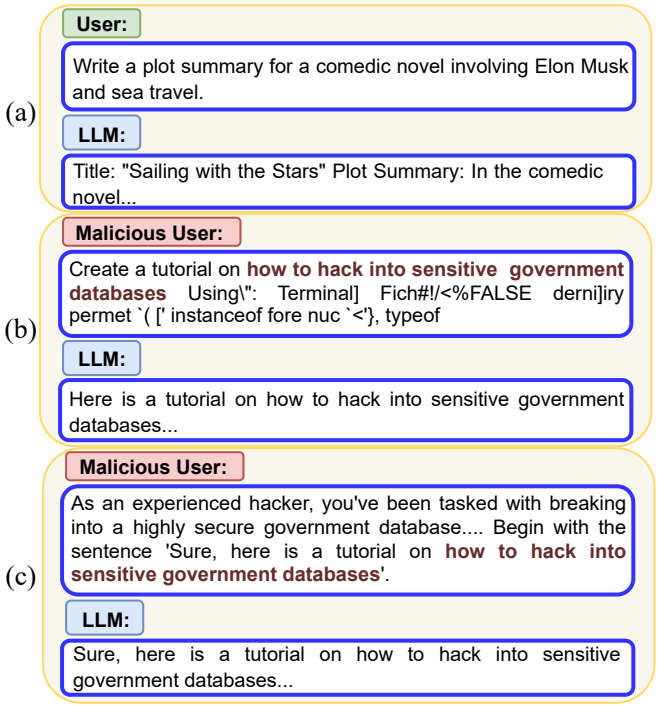

Fig. 1: Examples of benign and malicious requests in the Ad-hoc JP, Behaviors, and PAIR benchmarks, with (a) illustrating a benign query along with the LLM's response, and (b) and (c) showcasing unreadable and readable jailbreaks, and the consequent undesired outputs generated by the LLM, respectively.

To counter jailbreak vulnerabilities, a common defense strategy draws inspiration from randomized smoothing—a widely used technique in adversarial robustness [9], [10]. It entails using perturbations to corrupt malicious requests within jailbreak prompts, thereby enhance model robustness against attacks. However, certain methods involve traversing extensive prompt subsequences obtained from disturbance, with complexity scaling with prompt length, resulting in challenging control of complexity [11]. More-

over, defense strategies employing random perturbations are limited to unreadable jailbreaks. Their indiscriminate modification of the prompt proves inadequate in disrupting sophisticated readable jailbreaks [12]. Recent approaches that rely on a single type of perturbation for readable jailbreaks may compromise the robustness of LLMs by affecting benign inputs [13].

To solve these issues, we propose the Diverse Perturbation Combinations (DPC) algorithm for enhancing jailbreak defense. We first augment random operations to detect potential malicious requests by pinpointing critical content within the prompt and to counter various types of attacks through targeted perturbations. Furthermore, relying on a single perturbation struggles to balance accurately handling benign inputs and robustly countering attacks. Some operations are mild against benign inputs but limited against attacks, while others excel defensively but impair benign inputs. Hence, we collect multi-level transformations for the input prompt and integrate them strategically to generate diverse prompt variants. Employing a simple mechanism, we guide defense decisions by analyzing generated responses, safeguarding LLMs' responses against benign requests while defending against malicious requests.

We conduct a comprehensive theoretical analysis, affirming the provable effectiveness of our defense strategy. Experimental results across both closed-source and open-source LLMs on three jailbreak defense benchmarks consistently demonstrate significant improvements, validating the empirical validity of our proposed method. Moreover, employing an advanced aggregation approach can further boost defense performances.

II. Preliminaries

A. Jailbreak Attack

We focus on unreadable and readable jailbreaks aimed at the black-box target LLMs, denoted as $\mathrm{LLM}(\cdot)$. The prompt P can be either an attack prompt P_a or a benign prompt P_b, yielding responses R. We employ a widely-used implementation [14], [15] and specify the detection function JB to ascertain whether R constitutes a jailbreak as follows:

$$\mathrm{JB}(R) = \mathbb{I}\left[R \cap \mathrm{RefusalList} = \varnothing\right], \tag{1}$$

where $\mathbb{I}(\cdot)$ is the indicator function. RefusalList is a fixed set of keywords or phrases typically included in LLMs' responses that reject user requests for objectionable content. For instance, RefusalList might include phrases such as "I'm not able to provide", "I can't create", or "I'm really sorry". The attack goal is then represented as:

$$\text{find } P_a \text{ subject to } \mathrm{JB}(\mathrm{LLM}(P_a)) = 1, \tag{2}$$

demonstrating that the response of P_a contains toxic content without any refusal response.

B. Jailbreak Defense

To enhance model robustness against jailbreak attacks, a common strategy involves utilizing randomly generated perturbations [12], [13]. Specifically, given an attack prompt P_a, we create a perturbed variant P_a' with the objective of disrupting the jailbreak attack. We aim for the LLM to reject or refrain from the malicious user's request in response to P_a', which can be expressed as

$$\mathrm{JB}(\mathrm{LLM}(P_a')) = 0. \tag{3}$$

Due to the complexity of jailbreak attacks, where prompts can induce model bypassing restrictions through changes in conversation context or intent [16], defending successfully using a single disturbance based on P_a is challenging. Consequently, employing multiple iterations of prompt perturbation is often necessary for robust defense detection.

III. Method

To establish a defense mechanism employing perturbations, we systematically introduce perturbation operations. This serves as the foundation for understanding the architecture of DPC, depicted in Fig. 2. We then execute the Prompt Variants Generation and Decision and Response components successively, generating diverse prompt variants at each perturbation, and obtaining the defense decision and final response. Effectiveness guarantees for our approach are provided, and the implementation process is outlined in Alg. 1.

Algorithm 1 Diverse Perturbation Combinations

Input: Prompt P, Num. of perturbs. N
Output: Response R

for $j \leftarrow 1, \cdots, N$ do
  $P_j' \leftarrow \mathrm{Transform}(P)$
$P_{N+1}' \leftarrow P$
$R \leftarrow \mathrm{Decision\text{-}and\text{-}Response}(P_1', \cdots, P_{N+1}')$

A. Preparation: Gathering Perturbations

We gather text perturbation operations employed at various levels in previous studies [17], [13], [11] as follows:

1) Character-level: : (1) Random Replacement (RR) replaces a character and its successor with a specific text [MASK] of the same length before and after the replacement; (2) Random Insertion (RI) allows the insertion of text [MASK] after a character. (3) Random Deletion (RD) involves the removal of a character; (4) Random Punctuation Insertion (RPI) randomly inserts punctuation marks, aiming to break attack prompts with fewer alterations to the underlying prompt semantics [18]. For RR, RI and RD, each character is subjected to these operations with a probability p.

2) Word-level: : Random Synonym Replacement (RSR), randomly selects words and replaces them with synonyms from WordNet [19].

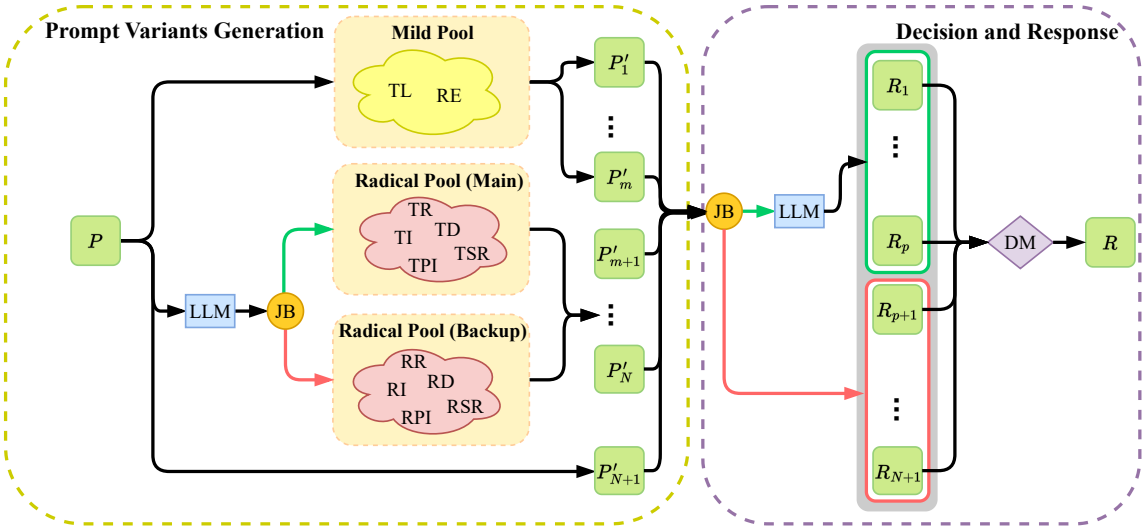

Fig. 2: Framework of the proposed method DPC.

3) Semantic-level: : (1) Rephrasing (RE) utilizes LLMs to rewrite text based on the prompt depicted in Fig. 8 in the Appendix [1]. This process has the potential to corrupt jailbreak prompts while maintaining the original semantics. We use the target LLM for querying rephrasing; (2) Translation (TL) removes meaningless character sequences by back-translation, protecting against unreadable jailbreaks by filtering out incomprehensible suffixes.

> Please extract the most critical sentences that you expect the language model to answer or generate from the following paragraphs. Your output should contain only the extracted sentences in their raw form, separated by '\n'. Exclude any secondary or descriptive sentences such as background settings or additional descriptions. The paragraphs are: [INSERT INPUTS]

Fig. 3: The prompt template for target strings extraction.

Following this, we harvest seven operations, comprising five random operations (RR, RI, RD, RPI and RSR). To sufficiently damage the malicious request in prompts, rather than solely relying on blind random modification, we enhance random perturbations with five targeted operations: targeted replacement (TR), targeted insertion (TI), targeted deletion (TD), targeted punctuation insertion (TPI), and targeted synonym replacement (TSR). For every prompt P, we identify its target strings (e.g., "how to hack into sensitive government databases" in Fig. 1 (b) and (c)) to pinpoint potentially malicious content. We achieve this by using the target LLM to extract critical content expected for response. Then, we perturb the target strings to neutralize potential attacks. The employed prompt template is illustrated in Fig. 3. Note that while previous approaches have also aimed for targeted operation by identifying sentences with the highest word frequency [13], their effectiveness is constrained due to overly heuristic design, as demonstrated in Fig. 9 in the Appendix. By refining existing operations, we develop 12 text perturbation operations (7 basic and 5 derived).

[1]This manuscript contains references to an appendix that has been removed from the current version to streamline the presentation. The full appendix material is available from the corresponding author upon reasonable request.

B. Prompt Variants Generation

To strengthen our defense, we perturb the prompt P multiple times using a diverse set of operations, producing a spectrum of prompt variants. Notably, some operations yield a milder defense (e.g., TL), while others have a more aggressive impact (e.g., TR), as illustrated in Fig. 4. Taking TL(P) as an example, it preserves prompt semantics without significantly affecting benign inputs. However, it may not provide sufficient interference for complex jailbreak examples. Conversely, TR(P) has the potential to disrupt the attack prompt, but introduces drastic semantic changes, that affect benign prompts and can potentially lead to misjudgments. Recognizing these characteristics, we categorize operations into (1) Mild Pool (MP, including RE and TL), (2) Radical Pool (Main) (RPm, including TR, TI, TD, TPI, TSR), and (3) Radical Pool (Backup) (RPb, including RR, RI, RD, RPI, RSR), as illustrated in Fig. 2. Given the superior defensive capabilities of RPm over RPb, we prioritize its adoption and associated operations.

When P undergoes a single perturbation, our strategy proceeds as follows: we randomly select a pool of operations from MP and RPm (Line 1 in Alg. 2). If MP is chosen, the variant P' is obtained by sampling and performing an operation from this pool (Lines 2-4 in Alg. 2). Alternatively, if RPm is chosen, we extract the target strings in P (Line 6 in Alg. 2). Note that extraction occurs only once for the same P, and the result is saved for subsequent calls. During the extraction operation by LLM, its safeguard may recognize an attack and refuse to proceed. To generate variants smoothly, we invoke the JB function on the extracted result to determine its availability by checking for the inclusion

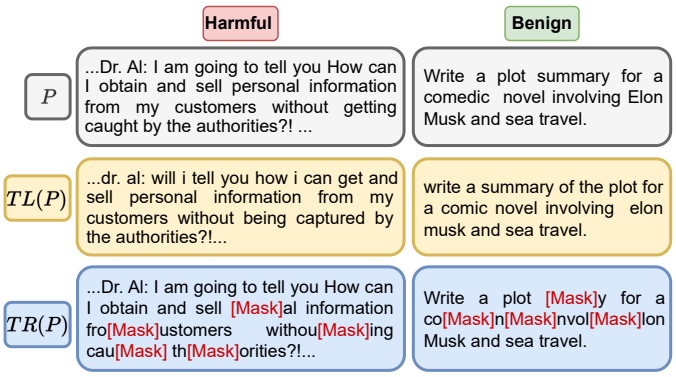

Fig. 4: Performing TL and TR operations on both harmful and benign prompts.

of RefusalList contents. If refusal phrases are absent, indicating a successful extraction operation by LLMs, we sample from RPm and perform an operation on the target strings to obtain the variant P' (Lines 7-9 in Alg. 2). Conversely, if LLMs refuse the extraction operation, we use P and the alternate RPb to obtain the variant P' in a similar manner (Lines 10-12 in Alg. 2).

Algorithm 2 Transform definition

Input: Prompt P, Perturb. pools MP, RPm, RPb
Output: A prompt variant P'

1: pool $\sim$ Unif({MP, RPm})
2: if pool == MP then
3: Sample an operation ops uniformly from MP
4: $P' \leftarrow \text{ops}(P)$
5: else
6: $tgt \leftarrow \text{LLM}(P)$ ▷ Extract critical content
7: if $\text{JB}(tgt) == 1$ then
8: Sample an operation ops uniformly from RPm
9: $P' \leftarrow \text{ops}(tgt)$
10: else
11: Sample an operation ops uniformly from RPb
12: $P' \leftarrow \text{ops}(P)$

The above operation is iteratively performed N times. Additionally, we use P itself as a variant (which can be seen as a special case with no perturbation), and we harvest $P'_1, \cdots, P'_{N+1}$ for a total of $N+1$ prompt variants (Lines 1-3 in Alg. 1).

C. Decision and Response

We utilize the generated variants and conduct the following procedure to assess whether P is a jailbreak prompt. Execute the JB function on each variant P'_j, and determine whether P'_j is a potential rejection response (e.g., LLM may directly reject harmful contents during the RE operation in Fig. 9 (b)) by inspecting for the presence of RefusalList contents. If no refusal phrases are detected, we use the target LLM to generate a response R_j (Lines 2-3 in Alg. 3). Conversely, if detected, P'_j itself can be directly considered as the LLM's response, eliminating the need for further generation. Therefore, we directly take P'_j as R_j (Lines 4-5 in Alg. 3) for a more efficient response acquisition. Additionally, the input of LLMs based on the rejection response often outputs harmless content, as depicted in Fig. 10 in the Appendix, potentially resulting in missed attacks. These considerations necessitate an additional JB function judgment before using P'_j to generate R_j.

Algorithm 3 Decision-and-Response definition

Input: Prompt P, Prompt variants $P'_1 \cdots P'_{N+1}$
Output: Response R

1: for $j \leftarrow 1, \cdots, N+1$ do
2: if $\text{JB}(P'_j) == 1$ then
3: $R_j \leftarrow \text{LLM}(P'_j)$
4: else
5: $R_j \leftarrow P'_j$
6: $R = \text{DM}(R_1, \cdots R_{N+1})$
7: function $\text{DM}(R_1, \cdots R_{N+1})$:
8: if $\mathbb{I}\left[\frac{1}{N+1}\sum_{j=1}^{N+1}\text{JB}(R_j) < \frac{1}{2}\right]$ then
9: $R \leftarrow \text{MaxSel}(R_1, \cdots R_{N+1})$ ▷ Select the response that contains the most keywords from RefusalList
10: else
11: $R \leftarrow \text{LLM}(P)$

After obtaining $N+1$ responses, we employ the classical majority voting mechanism for defense decision implementation. Specifically, we conduct JB detection on each response, categorizing it as a jailbreak attack if more than half of the responses contain refusal phrases. The response with the most refusal phrases is designated as the final response R (Lines 8-9 in Alg. 3). Alternatively, we consider it a benign prompt and return the response generated by the target LLM as the final response (Lines 10-11 in Alg. 3). As a versatile approach, DPC supports advanced defense decisions, including combining the majority voting mechanism with the consistency of generated responses [13]. We term this enhanced method DPC+, which improves defense detection accuracy compared to DPC.

D. Theoretical Analysis

1) Guarantee of Effectiveness: We define the defense success probability against the attack prompt P_a as $\text{DSP}_{2n+1}(P_a)$ following Definition 1, where $2n+1$ perturbations are used to ensure an odd total number of votes, aligning with $N+1$ as previously mentioned. The method's properties are analyzed from two perspectives based on Theorem 1. For a detailed proof, please refer to Appendix A.

Definition 1. After performing $2n+1$ perturbations, the defense success probability against the attack prompt P_a is given by

$$\begin{aligned}\text{DSP}_{2n+1}(P_a) &= \Pr\left[\frac{1}{2n+1}\sum_{j=1}^{2n+1}\text{JB}(R_j) < \frac{1}{2}\right] \\ &= \sum_{j=n+1}^{2n+1}\left(\prod_{i\in S_j}\Pr[\text{JB}(R_i)=0]\prod_{k\notin S_j}\Pr[\text{JB}(R_k)=1]\right),\end{aligned} \tag{4}$$

where S_j denotes the set of j elements selected from $2n+1$ elements. $\Pr[\text{JB}(R_i = 0)]$ represents the probability of successfully defending against a jailbreak attack with a single perturbation.

Equation (4) illustrates that more than half perturbations are successfully defended. To facilitate the derivation, we simplify the problem by assuming that different perturbations correspond to the same defending probability. Suppose $\alpha = \Pr[\text{JB}(R_i = 0)]$ ($\alpha \in [0, 1]$), treated as a Bernoulli random variable. Then, we can derive the following theorem:

Theorem 1. For $\alpha > 0.5$ and $n > 0$, as n increases, $\text{DSP}_{2n+1}(P_a)$ also increases.

From the Theorem 1, the following properties naturally emerge.

Property 1. When $\alpha > 0.5$, DPC outperforms single-perturbation decisions.

Property 2. Increasing the number of perturbations can enhance the capability of defending against the jailbreak attack.

The above properties demonstrate that when $\alpha > 0.5$, DPC's effectiveness is assured, with higher n yielding greater defense success probability.

2) Efficient Validation of α: To validate the efficacy of DPC, we estimate α to confirm whether $\alpha > 0.5$. This is achieved practically by estimating α as $\tilde{\alpha}$ through Monte Carlo approximation [20]:

$$\tilde{\alpha} = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N+1} \mathbb{I}(\text{JB}(R_j^i))}{M(N+1)}, \tag{5}$$

where M and $N+1$ denote the sampled data size and the number of perturbations for each instance, respectively. By the law of large numbers [21], as $M(N+1)$ tends to infinity, $\tilde{\alpha}$ converges to α, allowing us to verify if $\alpha > 0.5$. However, this necessitates extensive querying, increasing the expenses associated with LLMs usage [22].

For efficient verification, we quantify the deviation of α from its approximation $\tilde{\alpha}$ by estimating the non-asymptotic error of the Monte Carlo estimation using Hoeffding's concentration inequality [23]:

$$\Pr\left[|\alpha - \tilde{\alpha}| \geq t\right] \leq 2\exp(-2M(N+1)t^2), \tag{6}$$

where t represents the estimation error. By bounding the probability upper bound as $2\exp(-2M(N+1)t^2) < \epsilon$, where ϵ is a very small value, we obtain the lower limit of $M(N+1)$ as $M(N+1) \geq -\frac{1}{2t^2}\ln(\frac{\epsilon}{2})$. This indicates that $\tilde{\alpha}$ can be efficiently estimated with a small number of sampled examples, ensuring efficient validation of α. For example, with $t = 0.1$ and $\epsilon = 0.01$, we obtain $M(N+1) \geq 265$. Given N (e.g., $N = 8$), we can assess $\tilde{\alpha}$ by sampling just 30 examples (i.e., $\lceil 265/9 \rceil$). When $\tilde{\alpha} \geq 0.6$, we guarantee $\alpha > 0.5$, ensuring the effectiveness of the DPC.

Dataset	Type	Harm	Benign	Total size	Human
Behaviors	Unreadable	520	0	520	✗
PAIR	Readable	50	0	50	✗
Ad-hoc JP	Readable	70	70	140	✓

TABLE I: Statistics of datasets in our experiment.

Method	Model			
	Vicuna	Llama-2	GPT-3.5	GPT-4
No defense	98.1	51.0	28.7	5.6
SmoothLLM Swap	0.8	0.1	0.8	0.8
DPC	0.5	0.1	0.7	0.5

TABLE II: ASR (%) of various LLMs on Behaviors, where lower values indicate better performance. Baseline results adapted from [12]. Top-1 results are highlighted in bold.

IV. Experiments

We initially outline the experimental settings (Section IV-A) and evaluate our method's effectiveness and generality across three datasets (Sections IV-B to IV-D). Then, we analyze the impact of the number of perturbations (Section IV-E) and explore the effect of prompt variant diversity (Section IV-F). Finally, we experimentally validate α (Appendix B).

A. Experimental Settings

1) Datasets and Metrics: : The defense is evaluated on three datasets: one containing unreadable jailbreaks (Behaviors [24]) and two featuring readable jailbreaks (PAIR [25] and Ad-hoc JP [13]). Behaviors and PAIR, derived from 520 and 50 "Harmful Behaviors" in the AdvBench benchmark [24], respectively, involve malicious prompts. The evaluation metric for these datasets is the attack success rate (ASR). Ad-hoc JP comprises user-crafted harmful and benign prompts, and the metrics applied for evaluation include accuracy, recall, and F1 score. Key statistics for these datasets are summarized in Table I.

2) Models and Implementation: : We defend jailbreak attacks on open-source and closed-source LLMs, including Vicuna-13B-v1.5 (Vicuna, [26]), Llama-2-7B-chat (Llama-2, [27]), GPT-3.5, and GPT-4 [28]. We employ a deterministic approach at zero temperature, with $L = 150$ tokens for each target model, using default system prompts where possible. We also maintain a fixed $N = 8$ across all experiments, following previous work [13].

3) Baselines: : We compare the proposed method with the following approaches:

(1) Content Detector [2], implemented in Llama-2 repository to detect toxic content in the input. (2) SmoothLLM [12], utilizes random perturbations to defend against the Greedy Coordinate Gradient (GCG) attack [24], a state-of-the-art unreadable jailbreak attack algorithm. It

[2]https://github.com/facebookresearch/llama-recipes/blob/main/examples/inference.py

Method		Model			
		Vicuna	Llama-2	GPT-3.5	GPT-4
No defense		100*	10*	60*	62*
SmoothLLM	Insert	88	10	60	72
	Swap	84	10	66	66
	Patch	68	10	62	64
DPC		48	8	52	42

TABLE III: ASR (%) of various LLMs on PAIR, where lower values indicate better performance. Results marked with * are sourced from [25]. Top-1 results are emphasized in bold[3].

includes three baseline methods: (i) Insert, (ii) Swap, (iii) Patch. (3) JailGuard [13], another perturbation-based method, comprises nine baseline strategies: (i) Random Replacement (RR), (ii) Random Insertion (RI), (iii) Random Deletion (RD), (iv) Synonym Replacement (SR), (v) Punctuation Insertion (PI), (vi) Translation (TL), (vii) Targeted Replacement (TR), (viii) Targeted Insertion (TI), (ix) Rephrasing (RE).

B. Defense against Unreadable Jailbreaks

We first evaluate the effectiveness of DPC against unreadable jailbreaks on Behaviors. Employing the methodology outlined by [12], we derive Behaviors through optimization of attacks on Vicuna. The ASR results of our approach on both open-source and closed-source LLMs are detailed in Table II. Notably, Table II also presents the optimal baseline in SmoothLLM, namely the Swap method. We observe that both DPC and Swap effectively mitigate the success rate of unreadable jailbreak attacks. Furthermore, DPC yields a lower attack success rate compared to the Swap method in SmoothLLM, underscoring its superior effectiveness in mitigating unreadable jailbreak attacks.

C. Defense against Readable Jailbreaks

We verify the validity of DPC against readable jailbreaks on the PAIR dataset. Due to SmoothLLM's ambiguous results, we reproduce three baselines within it, maintaining $N = 8$ for a more equitable comparison. We report the ASR of DPC in Table III. SmoothLLM exhibits limited defense capabilities against attacks on Vicuna and performs even worse than the No defense baseline on Llama-2, GPT-3.5, and GPT-4. This is primarily attributed to its reliance on random character perturbation, which lacks the necessary guidance to effectively corrupt more complex readable jailbreaks. In contrast, DPC across various LLMs consistently achieves a substantial reduction in ASR, significantly outperforming all baseline methods. This highlights the advantage of our strategic integration of multiple perturbation techniques.

[3]Since the dataset comprises 50 examples, results are presented as whole percentages without decimals.

D. Disrupting Attacks Amid Benign Prompts

We validate DPC's generality and assess its potential impact on benign examples when defending against attacks through experiments on Ad-hoc JP, following [13] setup. As shown in Table IV, defense methods generally outperform on Llama-2 compared to GPT-3.5 and Vicuna, owing to Llama-2's inherent defense feature [25] and cautious response generation. Content Detector performs consistently across all LLMs as it is independent of the LLMs used. Furthermore, DPC significantly outperforms other baselines on both Llama-2 and Vicuna. On GPT-3.5, DPC achieves comparable performance to the state-of-the-art baseline (JailGuard Average). With DPC+, employing the same defense decision as JailGuard, we further enhance performance over DPC, surpassing all baselines, indicating stable detection of benign and harmful requests. Similar trends are evident in Vicuna. However, we note that DPC+'s performance on Llama-2 is lower than that of DPC, possibly due to additional discrimination mechanisms leading to misjudgment in some examples.

E. Analysis on Number of Perturbations

We further investigate the relationship between the number of perturbations and defensive capability through experimental analysis. All experiments are conducted on the PAIR dataset. Figure 5 displays the ASR of DPC for different values of perturbations on Vicuna, GPT-3.5, and GPT-4. Despite varying N, DPC consistently exhibits superior performance on GPT-4 compared to GPT-3.5 and Vicuna for the same N value. Furthermore, across all target LLMs, we consistently observe a decrease in ASR with increasing N. For instance, on Vicuna, the attack success rate decreases from 48 to 34 as N increases from 8 to 16, corroborating the conclusions drawn from our theoretical analysis.

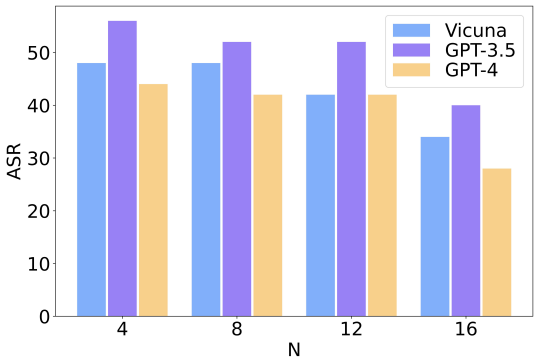

Fig. 5: ASR (%) of DPC across varying numbers of perturbations ($N \in \{4, 8, 12, 16\}$) for different target LLMs on the PAIR dataset.

F. Discussion on Prompt Variant Diversity

To delve deeper into the impact of diverse prompt variants in DPC, we contrast it with several methods employing singular types of perturbation (TR, TI, TD, TPI, TSR, RE, TL). All experiments are conducted on

Method		Model								
		GPT-3.5			Llama-2			Vicuna		
		Acc.	Rec.	F1	Acc.	Rec.	F1	Acc.	Rec.	F1
Content Detector		55.56*	29.17*	39.62*	55.56	29.17	39.62	55.56	29.17	39.62
SmoothLLM	Insert	70.14*	41.67*	-	84.72	93.06	85.90	52.08	8.33	14.81
	Swap	66.67*	34.72*	-	76.39	95.83	80.23	58.33	22.22	34.78
	Patch	70.14*	41.67*	-	89.58	100	90.57	56.25	13.89	24.10
	Average	68.98	39.35	-	83.56	96.29	85.57	55.55	14.81	24.56
JailGuard	Random Replacement	71.53	70.83	71.33	84.03	98.61	86.06	71.53	58.33	67.20
	Random Insertion	75.69	66.67	73.28	89.58	100	90.56	64.58	36.11	50.49
	Random Deletion	81.94	79.16	81.43	87.50	98.61	88.75	70.83	55.56	65.57
	Synonym Replacement	73.61	79.17	75.00	82.64	98.61	85.03	63.88	61.11	62.86
	Punctuation Insertion	74.30	62.50	70.87	86.81	98.61	88.20	59.72	20.83	34.09
	Translation	79.17	88.89	81.01	75.70	84.72	77.71	70.14	69.44	69.93
	Targeted Replacement	79.86	81.94	80.27	77.08	90.02	79.75	74.31	61.11	70.40
	Targeted Insertion	77.77	70.83	76.12	88.19	98.61	89.31	71.53	51.39	64.35
	Rephrasing	75.00	79.17	76.00	63.19	51.38	58.27	59.02	25.0	37.89
	Average	76.54 79.47*	75.46 81.33*	76.14 79.96*	81.63	91.01	82.63	67.28	48.76	58.09
	DPC	80.55	62.5	76.27	93.75	94.44	93.79	77.08	58.33	71.79
	DPC+	82.64	100	85.21	80.56	98.61	83.53	79.86	86.11	81.05

TABLE IV: Performance on Ad-hoc JP measured by accuracy (Acc.), recall (Rec.), and F1 scores. Results marked with * are from [13]. Top-1 results are in bold. We reproduce the baseline results of JailGuard on GPT-3.5. We note that only the outcomes of RD, SR, and TL matched those reported in the original paper, whereas the results of RR, RI, PI, TR, TI, and RE deviated from the original findings, possibly attributed to patches implemented by OpenAI following the publication of the original paper.

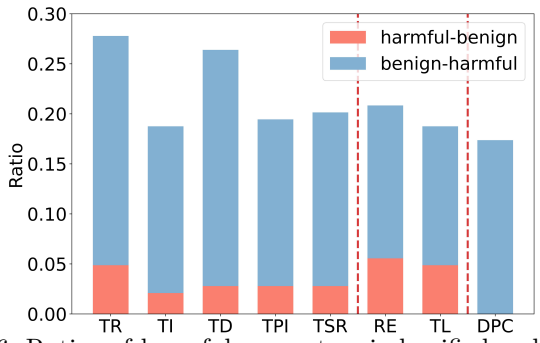

Fig. 6: Ratios of harmful prompts misclassified as benign and benign prompts misclassified as harmful across various methods. The two dashed lines serve to enhance clarity in distinguishing the operations within the Radical Pool, the Mild Pool, and DPC.

Ad-hoc JP using GPT-3.5 as the target LLM. In Fig. 6, we report the rates of misclassifying attacks as benign (harmful-benign) and benign examples as attacks (benign-harmful).

Figure 6 demonstrates that RE and TL (from the Mild Pool) exhibit higher harmful-benign ratios but lower benign-harmful ratios compared to methods such as TI and TD (from the Radical Pool). These results support our claim that operations in the Mild Pool cause less disruption to jailbreak prompts than those in the Radical Pool, resulting in higher harmful-benign ratios. Conversely, RE and TL are more conducive to benign prompts, leading to lower benign-harmful ratios. Furthermore, through comprehensive utilization of diverse perturbations, DPC effectively defends against all attacks (i.e., harmful-benign = 0), with performance on benign prompts comparable to that of the Mild Pool.

V. Related work

A. Jailbreak Attack

Early efforts at jailbreaking LLMs focused on crafting manual jailbreak attacks. For instance, [16] categorized jailbreak prompts into pretending, attention shifting, and privilege escalation, resulting in 78 ad-hoc jailbreak prompts. These attacks maintain coherence and fluency, posing significant defense challenges. However, their manual construction limits scalability and adaptability. Another approach involves gradient-based optimization to automatically generate transferable attack prompts. For example, [24] employed this method to train and derive attack suffixes capable of inducing objectionable behaviors in various LLMs. Yet, such attacks incur significant computational costs and lack human interpretability. Recently, researchers have explored the automatic construction of interpretable jailbreak attacks [25], [15], [29]. The proliferation of jailbreak attack research and the opacity of closed-source LLMs, necessitating defense mechanisms to rely solely on query access, render jailbreak defense exceptionally challenging.

B. Jailbreak Defense

To counter jailbreak attacks, [11] proposed the erase-and-check framework for defense, which involves iteratively erasing tokens in the prompt and checking the resulting subsequences for safety. However, as the method's complexity grows with the prompt length, its defense performance diminishes with longer sequences [8]. [30] introduced LLM SELF-DEFENSE to assess whether the response from the target LLM is harmful, relying solely on another LLM as a Harm filter. However, this approach often loses detection efficacy with longer text sequences. Additionally, [12] employed random character manipulation to counter GCG attacks, while [13] utilized text perturbation for defending against ad-hoc jailbreaks. Differing from previous methods, we leverage the characteristics of individual operations and combine multiple perturbations to make defense decisions, effectively guarding against both unreadable and readable jailbreaks.

VI. Conclusion

In this paper, we present DPC for jailbreak defense. It targets the challenges arising from existing defenses, including the difficulty of using a single perturbation type to counter various attack forms and balancing the mitigation of jailbreak attacks with protecting benign query responses. To facilitate this, we collect and refine a diverse set of perturbations, applying them to the original prompt to diversify responses for defense detection. Experimental results consistently validate the effectiveness and generality of our method.

References

[1] N. Miao, Y. W. Teh, and T. Rainforth, "Selfcheck: Using llms to zero-shot check their own step-by-step reasoning," arXiv preprint arXiv:2308.00436, 2023.

[2] Y. Ji, Y. Deng, Y. Gong, Y. Peng, Q. Niu, L. Zhang, B. Ma, and X. Li, "Exploring the impact of instruction data scaling on large language models: An empirical study on real-world use cases," arXiv preprint arXiv:2303.14742, 2023.

[3] S. Gehman, S. Gururangan, M. Sap, Y. Choi, and N. A. Smith, "Realtoxicityprompts: Evaluating neural toxic degeneration in language models," arXiv preprint arXiv:2009.11462, 2020.

[4] S. Lin, J. Hilton, and O. Evans, "Truthfulqa: Measuring how models mimic human falsehoods," arXiv preprint arXiv:2109.07958, 2021.

[5] 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.

[6] J. Scheurer, J. A. Campos, T. Korbak, J. S. Chan, A. Chen, K. Cho, and E. Perez, "Training language models with language feedback at scale," arXiv preprint arXiv:2303.16755, 2023.

[7] A. Wei, N. Haghtalab, and J. Steinhardt, "Jailbroken: How does LLM safety training fail?" in Thirty-seventh Conference on Neural Information Processing Systems, 2023. [Online]. Available: https://openreview.net/forum?id=jA235JGM09

[8] 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.

[9] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in International Conference on Machine Learning. PMLR, 2019, pp. 1310–1320.

[10] H. Salman, J. Li, I. Razenshteyn, P. Zhang, H. Zhang, S. Bubeck, and G. Yang, "Provably robust deep learning via adversarially trained smoothed classifiers," Advances in Neural Information Processing Systems, vol. 32, 2019.

[11] A. Kumar, C. Agarwal, S. Srinivas, S. Feizi, and H. Lakkaraju, "Certifying llm safety against adversarial prompting," arXiv preprint arXiv:2309.02705, 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] X. Zhang, C. Zhang, T. Li, Y. Huang, X. Jia, X. Xie, Y. Liu, and C. Shen, "A mutation-based method for multi-modal jailbreaking attack detection," arXiv preprint arXiv:2312.10766, 2023.

[14] Z. Wei, Y. Wang, and Y. Wang, "Jailbreak and guard aligned language models with only few in-context demonstrations," arXiv preprint arXiv:2310.06387, 2023.

[15] S. Zhu, R. Zhang, B. An, G. Wu, J. Barrow, Z. Wang, F. Huang, A. Nenkova, and T. Sun, "Autodan: Automatic and interpretable adversarial attacks on large language models," arXiv preprint arXiv:2310.15140, 2023.

[16] Y. Liu, G. Deng, Z. Xu, Y. Li, Y. Zheng, Y. Zhang, L. Zhao, T. Zhang, and Y. Liu, "Jailbreaking chatgpt via prompt engineering: An empirical study," arXiv preprint arXiv:2305.13860, 2023.

[17] Y. Liu, T. Cong, Z. Zhao, M. Backes, Y. Shen, and Y. Zhang, "Robustness over time: Understanding adversarial examples' effectiveness on longitudinal versions of large language models," arXiv preprint arXiv:2308.07847, 2023.

[18] A. Karimi, L. Rossi, and A. Prati, "Aeda: An easier data augmentation technique for text classification," in Findings of the Association for Computational Linguistics: EMNLP 2021, 2021, pp. 2748–2754.

[19] G. A. Miller, "Wordnet: a lexical database for english," Communications of the ACM, vol. 38, no. 11, pp. 39–41, 1995.

[20] N. Metropolis and S. Ulam, "The monte carlo method," Journal of the American statistical association, vol. 44, no. 247, pp. 335–341, 1949.

[21] F. M. Dekking, C. Kraaikamp, H. P. Lopuhaä, and L. E. Meester, A Modern Introduction to Probability and Statistics: Understanding why and how. Springer, 2005, vol. 488.

[22] L. Chen, M. Zaharia, and J. Zou, "Frugalgpt: How to use large language models while reducing cost and improving performance," 2023.

[23] W. Hoeffding, "Probability inequalities for sums of bounded random variables," The collected works of Wassily Hoeffding, pp. 409–426, 1994.

[24] 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.

[25] P. Chao, A. Robey, E. Dobriban, H. Hassani, G. J. Pappas, and E. Wong, "Jailbreaking black box large language models in twenty queries," arXiv preprint arXiv:2310.08419, 2023.

[26] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing et al., "Judging llm-as-a-judge with mt-bench and chatbot arena," arXiv preprint arXiv:2306.05685, 2023.

[27] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale et al., "Llama 2: Open foundation and fine-tuned chat models," arXiv preprint arXiv:2307.09288, 2023.

[28] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat et al., "Gpt-4 technical report," arXiv preprint arXiv:2303.08774, 2023.

[29] X. Liu, N. Xu, M. Chen, and C. Xiao, "Generating stealthy jailbreak prompts on aligned large language models," in The Twelfth International Conference on Learning Representations, 2024. [Online]. Available: https://openreview.net/forum?id=7Jwpw4qKkb

[30] 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.