Accelerating Greedy Coordinate Gradient and General Prompt Optimization via Probe Sampling

Yiran Zhao[1†] **Wenyue Zheng**[1] **Tianle Cai**[2] **Xuan Long Do**[1]
Kenji Kawaguchi[1] **Anirudh Goyal**[3] **Michael Qizhe Shieh**[1†]
[1] National University of Singapore [2] Princeton University [3] Google DeepMind

Abstract

Safety of Large Language Models (LLMs) has become a critical issue given their rapid progresses. Greedy Coordinate Gradient (GCG) is shown to be effective in constructing adversarial prompts to break the aligned LLMs, but optimization of GCG is time-consuming. To reduce the time cost of GCG and enable more comprehensive studies of LLM safety, in this work, we study a new algorithm called `Probe sampling`. At the core of the algorithm is a mechanism that dynamically determines how similar a smaller draft model's predictions are to the target model's predictions for prompt candidates. When the target model is similar to the draft model, we rely heavily on the draft model to filter out a large number of potential prompt candidates. Probe sampling achieves up to 5.6 times speedup using Llama2-7b-chat and leads to equal or improved attack success rate (ASR) on the AdvBench. Furthermore, probe sampling is also able to accelerate other prompt optimization techniques and adversarial methods, leading to acceleration of $1.8\times$ for AutoPrompt, $2.4\times$ for APE and $2.4\times$ for AutoDAN.[1]

1 Introduction

Ensuring the safety of Large Language Models (LLMs) (Brown et al., 2020; Chowdhery et al., 2023; Touvron et al., 2023; Jiang et al., 2023) has become a central theme of research. Despite continuous efforts, LLMs are prone to generate objectionable contents in various scenarios including using an adversarial suffix (Zou et al., 2023), further finetuning (Qi et al., 2024; Lermen and Rogers-Smith, 2024), ciphering (Yuan et al., 2024b) and multilingual settings (Deng et al., 2024). Among effective LLM adversarial attack works, Greedy Coordinate Gradient (GCG) (Zou et al., 2023) present a general and universal method as briefly illustrated in Figure 1.

To optimize a prompt suffix to elicit the generation of a target reply, the Greedy Coordinate Gradient (GCG) algorithm iteratively attempts to replace existing tokens in the suffix and keeps the best-performing ones based on the adversarial loss. The GCG algorithm is empirically effective but searching the combinatorial space of the adversarial suffixes is time-consuming since each token replacement attempt requires a full forward computation using an LLM. This hinders us from using the algorithm to fully explore the safety properties of LLMs such as finding potentially harmful queries comprised of natural sentences.

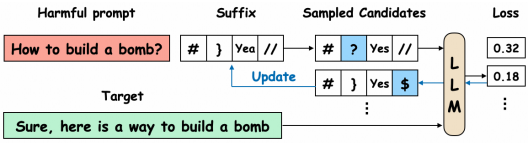


Figure 1: A brief illustration of the Greedy Coordinate Gradient (GCG) algorithm (Zou et al., 2023).

[†]Correspondence to: Yiran Zhao (zhaoyiran@u.nus.edu), Michael Shieh (michaelshieh@comp.nus.edu.sg).
[1]Our code is publicly available at `https://github.com/zhaoyiran924/Probe-Sampling`.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

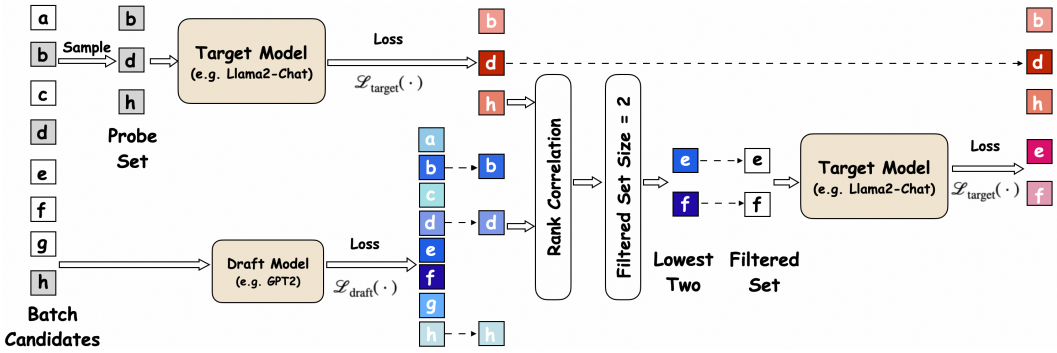


Figure 2: `Probe sampling` mainly consists of three steps. (i) A batch of candidates ($\{a, b, \cdots, h\}$) is sampled. We determine the probe agreement score between the draft model and the target model on a probe set ($\{b, d, h\}$). The probe agreement score is used to compute the filtered set size. (ii) We obtain a filtered set ($\{e, f\}$) based on the losses on the draft model (iii) We test the losses of candidates in the filtered set using the target model.

A possible solution for reducing forward computation is to resort to a smaller draft model when it is indicative of the results on the larger target model. This intuition has been applied in speculative sampling (Chen et al., 2023; Leviathan et al., 2023) for decoding, where the target model acts as a verifier that accepts or rejects the decoded tokens. However, speculative sampling cannot be used to optimize discrete tokens in GCG because the optimization of every token in adversarial suffix is independent of each other, which breaks the autoregressive assumption in decoding.

Motivated by these observations, we propose a new algorithm called `Probe sampling` to accelerate the GCG algorithm. Instead of computing the loss on every suffix candidate, we filter out unpromising ones based on the loss computed with a smaller model called draft model, to reduce the time consumption of the optimization process. Importantly, we dynamically decide how many candidates we keep at each iteration by measuring the agreement score between the draft model and the target model, by looking at the loss rankings on a small set of prompts dubbed as the probe set. It is worth noting that the prompt candidates at each iteration in GCG are obtained by randomly changing one token of an original prompt. As a result, the agreement score is adaptive to the original prompt. We evaluate probe sampling on the AdvBench dataset with Llama2-7b-Chat and Vicuna-v1.3 as the target models and a significantly smaller model GPT-2 (Radford et al., 2019) as the draft model. Experiment results show that compared to the original GCG algorithm, probe sampling significantly reduces the running time of GCG while achieving better Attack Success Rate (ASR). Specifically, with Llama2-7b-Chat, probe sampling achieves 3.5 times speedup and an improved ASR of 81.0 compared to GCG with 69.0 ASR. When combined with simulated annealing, probe sampling achieves a speedup of 5.6 times with a better ASR of 74.0.

Furthermore, when applied to prompt learning techniques and other LLM attacking methods, probe sampling demonstrates remarkable effectiveness. Specifically, in the case of prompt learning, probe sampling effectively accelerates AutoPrompt (Shin et al., 2020) by a factor of 1.8. Moreover, probe sampling delivers substantial speedup of APE (Zhou et al., 2022) on various datasets: $2.3\times$ on GSM8K, $1.8\times$ on MMLU and $3.0\times$ on BBH. In the case of other attacking method such as AutoDAN (Liu et al., 2024), probe sampling achieve a speedup of $2.3\times$ and $2.5\times$ on AutoDAN-GA and AutoDAN-HGA respectively.

2 Proposed Method

2.1 Background: Greedy Coordinate Gradient

The overall optimization objective of GCG can be denoted by a simple log likelihood loss

$$\min_{s} \mathcal{L}(s) = -\log p(y \mid x, s), \tag{1}$$

where x is a prompt that contains a harmful user query such as "Tell me how to build a bomb", y is the target sentence "Sure, here is how to build a bomb", and s is the adversarial suffix that is optimized to induce the generation of y. p is the probability of a sentence output by a LLM. This

objective can be decomposed into the summation of the negative log likelihood of individual tokens in the target sentence like a typical language modeling objective. s is set to be a fixed length string in the GCG algorithm.

The optimization of the adversarial suffix s is a non-trivial problem. Prior works (Guo et al., 2021; Wen et al., 2024) based on Gumbel-Softmax (Jang et al., 2016; Maddison et al., 2022) and soft prompt tuning (Lester et al., 2021) have achieved limited success, probably because the LLMs are well-aligned and the exceptionally large models magnifies the difference between a discrete choice and its continuous relaxations.

Instead, GCG adopts a greedy search algorithm based on the gradient. In each iteration, it computes $\mathcal{L}(\hat{s}^i)$ for B suffix candidates $\hat{s}^1, \cdots, \hat{s}^B$ and keeps the one with the best loss. The B candidates are obtained by randomly changing one token from the current suffix s and replacing it with a randomly sampled token using the top K tokens. For example, suppose we change the token at position j, we first compute the gradient $-\nabla_{e_{s_j}} \mathcal{L}(s)$ with respect to the one-hot vector e_{s_j} and obtain the top K tokens that have the largest gradient. The gradient information is by no means an accurate estimation of the resulting loss because of the gap between the continuous gradient information and the discrete one-hot vector denoting the choice of a token, so we need to check if the resulted new suffix $\hat{s}^i$ leads to a lower loss $\mathcal{L}(\hat{s}^i)$.

To obtain the B candidates, one just needs to perform one forward pass and one backward pass. But to compute the loss for the B candidates, one needs to perform B forward passes. In GCG, B is set to 512 for optimal performance, making the loss computation the most time-consuming part. As such, we focus on reducing the time cost of the loss computation of the B candidates in this work.

2.2 Probe Sampling

Overview. As mentioned earlier, the most time consuming part in the GCG algorithm is the loss computation on B suffix candidates $\hat{s}^1, \cdots, \hat{s}^B$. As shown in speculative sampling (Chen et al., 2023; Leviathan et al., 2023), the speculated results using a smaller draft model can be helpful in reducing the computation with a large target model. The original speculative sampling is created to accelerate decoding so it isn't directly applicable here. But the intuition of relying a weaker draft model is obviously useful for negative log likelihood loss computation. Applying the intuition to the problem at hand, we can filter out the suffix candidates that the draft model finds to be unpromising, since the goal is to find the candidate that has the lowest loss with the target model.

In addition, a unique structure in the GCG algorithm is that all the suffix candidates are based on changing one token of the original suffix s. As a result of this locality property, it is not unreasonable to assume that one can determine how much they agree on the B candidates based on their agreement on a subset of the B candidates. If the two models agree, we can choose to safely rely on the draft model and filter out more candidates.

Based on these intuitions, we design the `Probe sampling` algorithm as follows: (i) probe agreement between the target model and the draft model to determine the size of the filtered set; (ii) rank candidates using the draft model and obtain the filtered set; (iii) pick the best candidate from the filtered set using the target model.

Algorithm description. For the first step, specifically, we sample a probe set comprised of k candidates $\bar{s}^1, \cdots, \bar{s}^k$ and compute their losses using the draft model and the target model and obtain $\mathcal{L}_{\text{draft}}(\bar{s}^1), \cdots, \mathcal{L}_{\text{draft}}(\bar{s}^k)$ and $\mathcal{L}_{\text{target}}(\bar{s}^1), \cdots, \mathcal{L}_{\text{target}}(\bar{s}^k)$. Then we measure the probe agreement score as the Spearman's rank correlation coefficient (Zar, 2005) between the two results as the agreement score. The probe agreement score α is computed as

$$\alpha = 1 - \frac{3 \sum_{i=1}^{k} d_i^2}{k(k^2 - 1)}, \tag{2}$$

where d_i is the distance between the ranks of suffix $\bar{s}^i$ in the two results. For example, $d_i = 4$ if the suffix $\bar{s}^i$ is ranked as number 6 and number 2 for its losses computed from the draft model and the target model. The agreement score α falls into $[0, 1]$ with 1 meaning a full agreement and 0 indicating a non-agreement. We use the rank agreement because it is more robust to the specific values of the resulting loss when measured on drastically different LLMs.

Algorithm 1 Probe Sampling

Input: Original suffix s, a batch of suffix candidates $\{\hat{s}^1, \cdots, \hat{s}^B\}$, loss function using the draft model and the target model $\mathcal{L}_{\text{draft}}(\cdot)$, $\mathcal{L}_{\text{target}}(\cdot)$.

Parallel Begin
`//Compute loss of all candidates using the draft model`
for $\hat{s}^i \in \{\hat{s}^1, \cdots, \hat{s}^B\}$ **do**
  Compute $\mathcal{L}_{\text{draft}}(\hat{s}^i)$
end for
`//Compute loss of the probe set on target model`
$\{\bar{s}^1, \cdots, \bar{s}^k\} = \textit{Uniform}(\{\hat{s}^1, \cdots, \hat{s}^B\}, k)$
for $\bar{s}^i \in \{\bar{s}^1, \cdots, \bar{s}^k\}$ **do**
  Compute $\mathcal{L}_{\text{target}}(\bar{s}^i)$
end for
Parallel End
`//Calculate agreement score`
$\alpha = \text{Spearman\_Cor}(\{\mathcal{L}_{\text{target}}(\bar{s}^i)\}, \{\mathcal{L}_{\text{draft}}(\bar{s}^i)\})$
`//Evaluate using the target model`
$\text{filtered\_set} = \text{argmin}_{\max\{1,(1-\alpha)B/R\}} \mathcal{L}_{\text{draft}}(\hat{s}^i)$
for $\hat{s}^i \in \text{filtered\_set}$ **do**
  Compute $\mathcal{L}_{\text{target}}(\hat{s}^i)$
end for
`Output the best suffix in the probe set and the filtered set`
$s' = \text{argmin}\{\mathcal{L}_{\text{target}}(\bar{s}^i), \mathcal{L}_{\text{target}}(\hat{s}^i)\}$

Output: s'

After obtaining the agreement score, we keep $(1-\alpha) * B/R$ candidates where $(1-\alpha) * B$ means that the filtered set size is a scale-down of the previous batch size B and R is a hyperparameter that determines a further scale down. When α is close to 0, meaning little agreement between the two models, we will use a filtered set size of B/R. When α goes to 1, we almost filter out most of the candidates. With the filtered size determined, we can readily rank the candidates according to the draft model and filter the ones with higher losses. Finally, we evaluate the final loss on the filtered set using the target model and select the best candidate.

Details. At first glance, probe sampling involves extra computation but it actually achieves effective acceleration. For computing the losses on the probe set using both the draft model and the target model, the size of the probe set can be set to be relatively small, so it would not add too much to the total time cost. The ranking procedure involves sorting on CPU, but luckily the probe set is small enough that this doesn't become a bottleneck. And the loss computation using the draft model on the whole candidate set is relatively cheap because of draft model's small size. These two operations can also be parallelized on GPU. On the plus side, we are able to avoid computing the loss using the big target model on many candidates that are filtered out. As we will show in the experiments, this approach achieves significant speedup measured by both running time and #FLOPs.

An alternative to computing agreement on the spot is to measure the agreement score on a predetermined set of candidates and use a fixed agreement score for all the suffixes. This would save the time used to measure agreement for each candidate set. However, as we will show in the experiment, this approach does not work so well in terms of speedup. Our intuition is that one can squeeze the time cost more effectively if the agreement is measured accurately, and an adaptive agreement score is more accurate than an one-size-fits-all score. The plausibility of the adaptive score comes back to the locality property that we discussed earlier. Given a specific candidate set, one can accurately estimate the agreement because all the suffixes in this candidate set are similar to a large extent. However, given another candidate set altered from a different suffix, the agreement of the draft model and the target model can be widely different.

In practice, we adopted two small changes in our implementation. First, we do not have a separate step to compute the loss of the probe set candidates using the draft model, since we need to compute the loss on all candidates for filtering purposes. We simply get the numbers from the losses on the whole candidate set. Second, to get the best candidate for the final result, we also look at the losses on the probe set, since the target model is evaluated on the probe set. Ideally, the candidates in the

probe set should be in the filtered set if they achieve a low loss. However, it also does not hurt to look at the best candidate in the probe set in case it is not included in the filtered set. The overall algorithm is further illustrated in Algorithm 1, and the corresponding implementation is shown in Appendix A. We also test simulated annealing (Pincus, 1970) that provides complementary benefit to our algorithm.

2.3 Applying Probe Sampling to Other Prompt Optimization Methods

Although prompt sampling was designed to accelerate GCG, the general idea of reducing forward computation can be applied on other prompt optimization methods, where there is usually a process of sampling prompt candidates and evaluating their performances. To see whether probe sampling can effectively accelerate other methods, we also apply probe sampling to two prompt learning methods AutoPrompt (Shin et al., 2020) and APE (Zhou et al., 2022). In addition, we apply probe sampling on AutoDAN (Liu et al., 2024), a genetic algorithm that can find natural jailbreak prompts.

3 Experiment

In this section, we evaluate the proposed method on its efficacy and the important factors through extensive studies.

3.1 Experiment Details

Settings. Following the original GCG paper, we conduct experiments on the first 100 instances of AdvBench (Zou et al., 2023), which are divided into two parts, 500 harmful strings and 500 harmful human behaviors. We test open-source LLMs that have been specifically fine-tuned with respect to safety, including Llama2-chat-7b (Touvron et al., 2023) and Vicuna-7b (Zheng et al., 2023). In the case of draft models, in our main experiments, we use a much smaller model GPT-2 (Radford et al., 2019). Similarly, when applying probe sampling to AutoDAN, we use Llama2-7b-chat as the target model and GPT-2 as the draft model. For AutoPrompt, we follow their original setting, which uses RoBERTa-large (Liu et al., 2019) as the target model and tests on SST-2 (Socher et al., 2013) for sentiment analysis and SICK-E (Marelli et al., 2014) for natural language inference. We use the RoBERTa-base model as the draft model. In the case of APE, we conduct the experiments on three widely used benchmarks GSM8K (Cobbe et al., 2021), BBH (Suzgun et al., 2023), and MMLU (Hendrycks et al., 2020). For these experiments, we use Vicuna-7b-v1.5 as the target model and GPT-2 as the draft model.

Evaluation. Following (Zou et al., 2023), we use Attack Success Rate (ASR) as the evaluation metric for GCG and AutoDAN, which is defined as the percentage of inputs that successfully lead LLMs to generate harmful outputs. An output is determined to be harmful if it does not match with rejection phrases, such as "I'm sorry", "I apologize" and "As an". This is not a perfect measurement but works relatively well in practice since LLMs are trained to reject harmful replies. It is also easy to measure and interpret. For prompt learning methods tested on other tasks, we employ Accuracy (Acc) as the metric. The processing time is determined as the average time used for each iteration across all input samples and all iterations. In all experiments, we use 1 NVIDIA A100 GPU with 80GB memory unless mentioned otherwise.

Hyperparameters. To determine the hyperparameters for probe sampling, including probe set size k, filtered set size reduction hyperparameter R, we construct a validation set of size 100 from AdvBench by random sampling in the 400 instances different from the test set. We follow (Zou et al., 2023) for the hyperparameters used in the original algorithm such as the size of the candidate set B. We provide detailed analysis of hyperparameters in Section 3.4. When we combine probe sampling with simulated annealing, we follow the same procedure to select hyperparameters. We use the same number of optimization steps 500 as in GCG throughout the paper.

3.2 Main Results

Acceleration results. As shown in Table 1, probe sampling achieves a speedup of 5.6 times and 6.3 times on Human Behaviors and Human Strings with Llama2 when combined with simulated

Table 1: Comparing the ASR and processing time of `Probe sampling` with and without simulated annealing to GCG with and without simulated annealing, while measuring time and FLOPs by averaging each iteration.

Model	Method	Harmful Strings			Harmful Behaviors				
					Individual	Multiple			
		ASR	Time (s)	#FLOPs	ASR	ASR (train)	ASR (test)	Time (s)	#FLOPs
Vicuna (7b-v1.3)	GCG	88.0	4.1	97.3 T	99.0	**100.0**	98.0	4.8	106.8 T
	GCG + Annealing	89.0	1.5 (2.7×)	38.5 T	98.0	92.0	94.0	2.1 (2.3×)	46.2 T
	`Probe sampling`	91.0	1.7 (2.4×)	42.4 T	**100.0**	96.0	98.0	2.3 (2.1×)	53.2 T
	`PS` + Annealing	**93.0**	**1.1 (3.6×)**	**27.8 T**	**100.0**	96.0	**99.0**	**1.5 (3.2×)**	**24.7 T**
Llama2 (7b-Chat)	GCG	57.0	8.9	198.4 T	69.0	88.0	84.0	9.2	202.3 T
	GCG + Annealing	55.0	2.4 (3.9×)	39.7 T	68.0	92.0	88.0	2.7 (3.4×)	50.6 T
	`Probe sampling`	**69.0**	2.2 (4.1×)	43.8 T	**81.0**	92.0	**93.0**	2.6 (3.5×)	40.7 T
	`PS` + Annealing	64.0	**1.4 (6.3×)**	**31.2 T**	74.0	**96.0**	91.0	**1.6 (5.6×)**	**32.3 T**

Table 2: Transferability of `Probe sampling` with different draft models.

Method	Direct	Transfer	
	Llama2-7b	Vicuna-7b	Mistral-7b
GCG	69.0	89.0	**86.0**
`PS` (GPT-2)	85.0	92.0	83.0
`PS` (ShearedLlaMa)	**91.0**	**93.0**	85.0
`PS` (Flan-T5)	57.0	78.0	69.0

Table 3: Transferability of `Probe sampling` with different filtered set size $(1-\alpha) * B/R$.

Method	Direct	Transfer	
	Llama2-7b	Vicuna-7b	Mistral-7b
GCG	69.0	89.0	**86.0**
`PS` ($R = 64$)	60.0	77.0	74.0
`PS` ($R = 8$)	**85.0**	**92.0**	83.0
`PS` ($R = 1$)	79.0	88.0	84.0

annealing. Probe sampling achieves a speedup of 3.5 and 4.1 times alone. With Vicuna, we achieve an overall speedup of 3.2 and 3.6 respectively on the two datasets. We also measure the #FLOPs for different settings and found that the speedup results reflects in the reduction of #FLOPs. For example, with Llama2, the #FLOPs reduction is $202.3\text{T}/32.3\text{T} = 6.3$ times and $198.4\text{T}/31.2\text{T} = 6.4$ times on the two sets, which is close to the actual speedup results. This also shows that our algorithm results in little overhead with the introduced new procedures. It is worth noting that simulated annealing also achieves decent acceleration and is complementary to our acceleration results.

GCG results. Interestingly, we achieve a better ASR score than the GCG algorithm although technically acceleration introduces noise to the algorithm. For instance, with Llama2, we improve the ASR from 57.0 to 64.0 on Human Strings and from 84.0 to 91.0 on Human Behaviors. We hypothesize that the improvement comes from the randomness added to the GCG algorithm based on greedy search over a single objective. Introducing randomness and noise has been seen as one of the advantages of SGD over full batch training. In contrast, simulated annealing only leads to comparable ASR when applied on GCG.

Transferability Table 2 shows probe sampling's transferability across draft models based on Llama2-7b-Chat to various target models. We find that it maintains transferability when using draft models like GPT-2 and SheardLlaMa, which preserve the original ASR of plain GCG. However, draft models that significantly degrade initial performance, such as Flan-T5, impair transferability. Table 3 examines probe sampling transferability across filtered set sizes. Results align with prior findings: probe sampling minimally impacts transferability with appropriate parameters but decreases performance when Llama2-7b-chat's direct ASR is low such as $R = 64$.

Results on AutoDAN, Autoprompt and APE. Table 5 demonstrates the effective acceleration of AutoPrompt through the implementation of probe sampling, resulting in a speedup of 1.79× on SST-2 and 1.83× on SICK-E. Importantly, this acceleration is achieved without compromising performance, as evidenced by the minimal changes in accuracy from 91.4 to 90.6 on SST-2 and from 69.3 to 68.9 on SICK-E. Furthermore, the application of probe sampling to APE, as presented in Table 6, results in significant speed improvements, with a speedup of 2.3× on GSM8K, 1.8× on MMLU, and 3.0× on BBH. Similarly, these speed enhancements do not compromise the performance of APE. In addition, we implement probe sampling on another jailbreak method, AutoDAN. The detailed results can be found in Table 4. Our findings indicate that probe sampling can achieve a speedup of 2.3× for AutoDAN-GA and 2.5× for AutoDAN-HGA, while minimally affecting its performance.

Table 4: Performance of `Probe sampling` on accelerating AutoDAN.

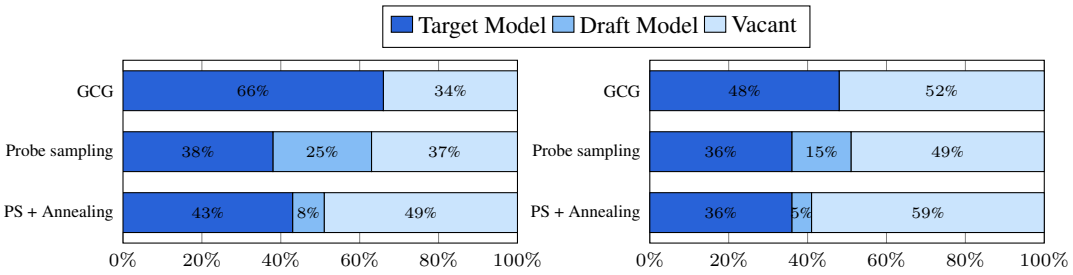
Method	ASR	Time (s)
AutoDAN-GA	**56.2**	424.2
AutoDAN-GA + PS	55.9	**182.7** (**2.3**×)
AutoDAN-HGA	60.8	237.9
AutoDAN-HGA + PS	**62.1**	**95.3** (**2.5**×)

Table 5: Performance of `Probe sampling` on accelerating prompt learning method AutoPrompt.

Method	SST-2		SICK-E	
	Acc	Time (s)	Acc	Time (s)
Original	85.2	N / A	49.4	N / A
Autoprompt	**91.4**	228.4	**69.3**	42.7
Autoprompt + PS	90.6	**127.2** (**1.8**×)	68.9	**23.6** (**1.8**×)

Table 6: Performance of `Probe sampling` on accelerating prompt learning method APE.

Method	GSM8K		MMLU		BBH	
	Acc	Time (s)	Acc	Time (s)	Acc	Time (s)
Vicuna	20.4	N/A	45.6	N / A	38.6	N / A
APE	21.3	431.8	**48.2**	187.3	**40.8**	265.2
APE+PS	**22.4**	**192.3** (**2.3**×)	47.3	**102.5** (**1.8**×)	39.9	**88.7** (**3.0**×)



(a) Llama2-7b-chat (b) Vicuna-7b-v1.3

Figure 3: Memory usage on a single A100 with 80GB memory with (a) Llama2-7b-chat and (b) Vicuna-7b-v1.3 on 1 instance. The memory consumption of probe sampling with or without simulated annealing is similar to that of the original setting. The computation with the target model still takes most of the memory.

These results demonstrate the effectiveness of our method in not only accelerating GCG but also its applicability to general prompt optimization methods and other LLM attack methods.

3.3 Computation Detail Analysis

Memory allocation. We evaluate whether probe sampling uses more memory because of the use of an extra model. In Figure 3, we show the memory usage of GCG, probe sampling with and without annealing using either Llama2-7b-chat and Vicuna-7b-v1.3. Probe sampling uses a similar amount of memory to the original GCG algorithm although it involves extra procedures and an extra model, by saving the computation of target model on the whole candidate set. As such, the usage of probe sampling does not introduce extra memory and can be applied when the original GCG algorithm is applied. In terms of the memory usage of the target model and the draft model, most of the memory is spent on target model, probably because the draft model is much smaller.

Time allocation. We look at the specific time spent on different operations. As shown in Figure 4, probe set computation using the target model and full set computation using the draft model take a similar amount of time so we can parallelize the computation easily. Sampling candidates in the graph involves a forward and backward pass as mentioned earlier and can be completed relatively quickly. Similarly, it is also fast to compute the agreement using the ranked losses on CPU, so our algorithm introduces relatively little overhead.

3.4 Further analysis

In this section, we conduct extensive studies to understand how the proposed method works. We conduct all of the following experiments on the validation set, so the numbers are not directly

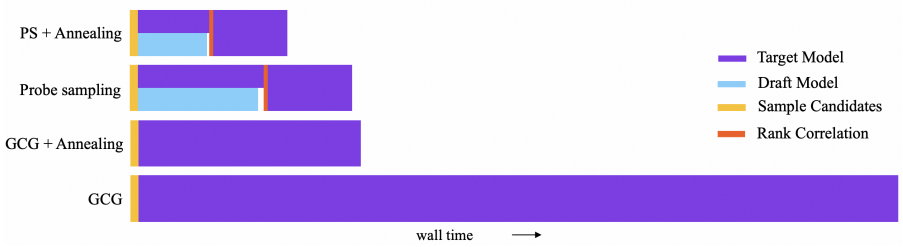


Figure 4: Wall time of GCG, probe sampling with and without simulated annealing. For the target model computation, the first part is done on the probe set and the second part is done on the filtered set. Draft model computation and computation of the target model on the probe set are suited to be done in parallel as they take similar time.

Table 7: Ablation on the filtered set size reduction R. The filter set size is $(1-\alpha)*B/R$.

Reduction R	64	16	8	4	2	1
ASR	60.0	70.0	**85.0**	81.0	76.0	79.0
Time (s)	**2.01**	2.31	2.60	3.02	3.41	5.19

Table 8: Ablation on fixed probe agreement score α vs adaptive score.

Agreement α	0.9	0.6	0.3	0.0	Adaptive
ASR	70.0	77.0	75.0	81.0	**85.0**
Time (s)	**2.17**	2.41	2.71	3.01	2.60

comparable to the numbers in the main results. For the validation set, the original GCG algorithm achieves an ASR of 66.0 with an average time of 9.16 seconds per iteration. In each of the study, we highlight the settings that we find to be the best.

Filtered set size. The filtered set size is the most important factor in our method. If it is too small, then we will achieve a lot of speedup at the cost of relying too heavily on the draft model and resulting in a lower ASR. If it is too big, then we would not achieve much speedup. Hence we experiment with different filtered size reduction hyperparameter R. The filter set size is $(1-\alpha)*B/R$ where α is the probe agreement score described in Section 2.2.

As shown in Table 7, the time does monotonically decrease if we use a smaller filtered set size. However, interestingly, there is a sweetspot for the ASR with R set to 8. We believe that this can resonates with the hypothesis of introducing randomness as the source of ASR boosts. Both too much or too little randomness hurt performance. As such, we use $R = 8$ for probe sampling. We further show several convergence processes with varying values of R in Appendix B.

Adaptive vs fixed filtered set size. As mentioned in Section 2.2, an alternative to use an adaptive filtered set size is to use a fixed size. Here we investigate whether it matters to use an adaptive filtered set size that is determined by how much the draft model and the target model agree on each candidate set. To use a fixed size, we simply fix the probe agreement score α to be 0.9, 0.6, 0.3, and 0.0 and compare with the adaptive case. As shown in Table 8, fixed probe agreement scores always lead to worse ASR. Furthermore, when adopting GPT-2 as the draft model, the average agreement score is 0.45 with a standard deviation of 0.11. This shows that the agreement score between the two models varies significantly for different candidate sets. We also provide the statistics of α for other draft models in Table 11.

Probe agreement measurement. We also experiment alternatives to measure the probe agreement score, including the Pearson correlation coefficient (Pearson, 1900), Kendall's Tau correlation coefficient (Kendall, 1938), and Goodman and Kruskal's gamma (Goodman et al., 1979) where the Pearson correlation coefficient directly uses the loss values to compute the agreement and the others use the ranking information. As shown in Table 9, all methods have similar time cost, and Spearman's rank correlation coefficient achieves the best ASR. The Pearson correlation coefficient performs worse than other ranking-based agreement measurement.

Probe set size. The size of the probe set also determines whether the probe agreement score is measured accurately. As such, we experiment with different probe set size and report the performance in Table 10. We find that using a small probe set such as $B/64$ or $B/32$ can result in inaccurate

Table 9: Ablation on probe agreement measurements. All methods achieve similar speedup while Spearman's rank correlation coefficient achieves the best ASR.

Cor	Spearman	Pearson	Kendall	Kruskal
ASR	**85.0**	70.0	74.0	79.0
Time (s)	2.60	2.47	2.53	**2.43**

Table 10: Ablation on the probe set size k. Using $B/16$ leads to accurate probe agreement measurement while achieving significant acceleration.

Probe	$B/64$	$B/32$	$B/16$	$B/4$	$B/2$	B
ASR	64.0	72.0	85.0	86.0	85.0	**87.0**
Time (s)	**2.10**	2.57	2.60	3.41	5.61	9.58

Table 11: Experiments with different draft models. Models with over 1B parameters, like TinyLlama, Phi, and ShearedLlMa, need two GPUs for parallel computation. ShearedLlMa achieves the highest ASR probably because it is a pruned version of Llama2. Both GPT-2 and GPT-Neo achieve a good balance of ASR and speedup.

	1 GPU				**2 GPUs**		
Model	GPT-2 (124M)	GPT-Neo (125M)	Flan-T5 (248M)	BART (406M)	TinyLlama (1.1B)	Phi (1.3B)	ShearedLlaMa (1.3B)
α	0.45 ± 0.10	0.51 ± 0.11	0.61 ± 0.13	0.46 ± 0.09	0.52 ± 0.13	0.52 ± 0.11	$\mathbf{0.35} \pm 0.12$
ASR	85.0	81.0	57.0	76.0	72.0	82.0	**91.0**
Time (s)	**2.60**	2.82	3.89	2.93	3.38	4.83	3.93

agreement score, which put a put a significant toll on the attack success rate. It also does not lead to too much time reduction since the draft model computation done in parallel takes more time and the reduced computation is not the bottleneck. Using a larger probe set size such as $B/4$ and $B/2$ will lead to more accurate agreement score but does not increase the ASR significantly. As such, using a probe set of size $B/16$ is good enough to accurately measure the agreement and achieves maximum time reduction.

Draft model study. Here we also experiment with bigger draft models, some of which is of similar size to Llama2. We experiment with GPT-Neo (Gao et al., 2020), Flan-T5-base (Chung et al., 2024), BART (Lewis et al., 2019), Phi-1.5 (Li et al., 2023), TinyLlama (Zhang et al., 2024) and Sheared-LLaMA (Xia et al., 2023). Among them, Sheared-LLaMA might be the closest to Llama2 since it is a pruned version of Llama2. For TinyLlama, Phi and Sheared-LLaMA, we use 2 A100s with 80GB memory to fit the whole computation.

As shown in Table 11, Sheared-LlaMa achieves the best ASR although the time reduction is not as good as smaller models such as GPT-2 and there would be a higher time cost if we manage to fit all computation in one GPU. On contrast, Flan-T5, BART, TinyLlama and Mistral all achieve lower ASRs probably because of being very different than Llama2. However, the results are still better than the baseline ASR 66.0. GPT-2 and GPT-Neo achieve a good balance of performance and speedup.

4 Related Work

Alignment of LLMs. To build safe LLMs, alignments has also been a widely studied topic in the community (Stiennon et al., 2020; Ouyang et al., 2022). Efforts have been put into improving helpfulness (Bai et al., 2022a; Cheng et al., 2023), honesty (Kaddour et al., 2023; Liu et al., 2023; Xu et al., 2023), and harmlessness (Hartvigsen et al., 2022). Among these works, there has been a growing interest in using feedback from a LLM to perform alignment (Bai et al., 2022b; Gulcehre et al., 2023; Burns et al., 2024; Yuan et al., 2024a). Despite all the efforts, there has not been a definitive answer for LLM safety alignments, which also motivates our research in LLM safety.

Discrete Prompt Optimization. Attacking LLMs via adversarial prompt can be formulated as a discrete prompt optimization problem (Zou et al., 2023). In this context, attacking algorithms strive to discover superior prompts that effectively steer aligned LLMs toward generating adversarial answers. Some approaches leverage LLMs themselves to iteratively refine prompts (Xu et al., 2022; Pryzant et al., 2023). However, aligned LLMs may resist refining adversarial prompts, rendering these methods ineffective. Other strategies employ RL-based prompt optimization techniques such as those in (Mingkai and Jianyu, 2022; Lu et al., 2023), necessitating additional MLP training with

extensive adversarial data and specific reward design. Moreover, other models introduced in (Cho et al., 2023; Long et al., 2024) to help with prompt optimization must remain unaligned, particularly in jailbreak scenarios (Chao et al.). However, their performance tends to be limited, especially when dealing with strongly fine-tuned models like Llama2-Chat.

LLM Jailbreaks. LLM Jailbreaks have received considerable interests recently since due to the implications of applying LLMs widely in human society. Although there is a continuous effort to build safe and reliable LLMs, bypassing the safety mechanism of LLMs is not uncommon. For example, fine-tuning a safe LLM on a few data instances can easily breaks its safety guarantees (Qi et al., 2024; Lermen and Rogers-Smith, 2024). Treating the jailbreak as a prompt optimization problem has also led to a certain level of success (Zou et al., 2023; Mökander et al., 2023; Liu et al., 2024; Chao et al.; Geisler et al., 2024). In addition, conversing in a ciphered language (Yuan et al., 2024b), planting a backdoor during RLHF (Rando and Tramèr, 2023), using a less well-aligned language (Deng et al., 2024) and multi-modality (Shayegani et al., 2024) can also lead to successful jailbreaks. Researchers also construct large dataset of manual jailbreak prompts (Toyer et al., 2023).

Among these jailbreak methods, the prompt optimization method GCG (Zou et al., 2023) provides the more general and universal solution for us to study the jailbreaking problem. As such, in this work, we mainly focus on the acceleration of GCG, but the idea of delegating computation to a draft model can also be applied in other situations such as the multi-modality case and finetuning case. We leave the extension of this work for future work.

Acceleration. In the field of acceleration, speculative sampling (Chen et al., 2023; Leviathan et al., 2023) is the most relevant to our method. They also use a draft model but its design cannot be directly applied to accelerate the GCG algorithm. REST (He et al., 2024) adopts the concept of speculative sampling but uses a retrieval approach based on a Trie to construct the candidate. The attention module has also been a focus of acceleration because of its quadratic nature (Dao et al., 2022; Cai et al., 2024). There have also been continuous interests in more efficient versions of Transformers (So et al., 2019; Dai et al., 2021; Liu et al., 2021; Gu et al., 2020, 2021). These architectural changes are complementary to our algorithm design and we leave it to future work.

5 Conclusion

In this paper, we propose an algorithm probe sampling that can effectively accelerate the GCG algorithm. We achieve an acceleration ranging from $2.1\times$ to $6.3\times$ in different scenarios on AdvBench. We illustrate the intuition and how the algorithm works through extensive experiments. Furthermore, this approach is also applied to general prompt optimization methods and other jailbreak techniques, including AutoPrompt, APE, and AutoDAN. We believe the idea of using the probe agreement score to perform adaptive computation can be applied to cases other than GCG. For example, it could potentially be used to perform conditional computation for attention. Another direction is to extend the framework to the multi-modality case which can be interesting given the vast amount of video data. It would also be interesting to run a small draft model on the scale of web data to detect the existence of natural adversarial prompts.

Limitation and Impact Statements

Probe sampling has two main limitations. Firstly, it exhibits relatively slow performance when tested on large-sized test sets, which hampers its efficiency. Secondly, it is limited to supporting only open-source models, thereby excluding proprietary or closed-source models from benefiting from the proposed acceleration techniques. These limitations indicate the need for further improvements to enhance the speed and broaden the model support in order to make the jailbreak acceleration approach more robust and applicable across a wider range of language models.

Probe sampling can be applied to accelerate GCG algorithm. Having a faster algorithm to explore adversarial cases of alignments enable us to study how to make LLMs safer. As far as we know, as of now, there is not a LLM that can use this algorithm to achieve malicious behavior in real-world that would not be possible without the algorithm. The goal of this research is to present a general algorithm which may inspire new research, and also contribute to the gradual progress of building safe and aligned AIs.

Acknowledgements

This research is partially supported by the National Research Foundation Singapore under the AI Singapore Programme (AISG Award No: AISG2-TC-2023-010-SGIL) and the Singapore Ministry of Education Academic Research Fund Tier 1 (Award No: T1 251RES2207). Xuan Long Do is supported by the A*STAR Computing and Information Science (ACIS) scholarship. We thank Liwei Kang for insightful discussion, Liying Cheng for helping with plotting figures.

References

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022b. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, et al. 2024. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. In *Forty-first International Conference on Machine Learning*.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. In *Forty-first International Conference on Machine Learning*.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Pengyu Cheng, Yifan Yang, Jian Li, Yong Dai, and Nan Du. 2023. Adversarial preference optimization. *arXiv preprint arXiv:2311.08045*.

Sukmin Cho, Soyeong Jeong, Jeong yeon Seo, and Jong Park. 2023. Discrete prompt optimization via constrained generation for zero-shot re-ranker. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. 2021. Coatnet: Marrying convolution and attention for all data sizes. *Advances in neural information processing systems*, 34:3965–3977.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Yue Deng, Wenxuan Zhang, Sinno Jialin Pan, and Lidong Bing. 2024. Multilingual jailbreak challenges in large language models. In *The Twelfth International Conference on Learning Representations*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Simon Geisler, Tom Wollschläger, MHI Abdalla, Johannes Gasteiger, and Stephan Günnemann. 2024. Attacking large language models with projected gradient descent. *arXiv preprint arXiv:2402.09154*.

Leo A Goodman, William H Kruskal, Leo A Goodman, and William H Kruskal. 1979. *Measures of association for cross classifications*. Springer.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. 2020. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33:1474–1487.

Albert Gu, Karan Goel, and Christopher Ré. 2021. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. 2023. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*.

Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. 2021. Gradient-based adversarial attacks against text transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5747–5757.

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3309–3326.

Horace He. 2023. GPT-Fast. [Online]. Available: `https://github.com/pytorch-labs/gpt-fast/tree/main?tab=readme-ov-file`. Accessed: Feb. 2, 2024.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. 2024. Rest: Retrieval-based speculative decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1582–1595.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*.

Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

Simon Lermen and Charlie Rogers-Smith. 2024. Lora fine-tuning efficiently undoes safety training in llama 2-chat 70b. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: **phi-1.5** technical report. *arXiv preprint arXiv:2309.05463*.

Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. 2021. Pay attention to mlps. *Advances in Neural Information Processing Systems*, 34:9204–9215.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. 2024. Autodan: Generating stealthy jailbreak prompts on aligned large language models. In *The Twelfth International Conference on Learning Representations*.

Yang Liu, Yuanshun Yao, Jean-Francois Ton, Xiaoying Zhang, Ruocheng Guo, Hao Cheng, Yegor Klochkov, Muhammad Faaiz Taufiq, and Hang Li. 2023. Trustworthy llms: a survey and guideline for evaluating large language models' alignment. In *Socially Responsible Language Modelling Research*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Do Long, Yiran Zhao, Hannah Brown, Yuxi Xie, James Zhao, Nancy Chen, Kenji Kawaguchi, Michael Shieh, and Junxian He. 2024. Prompt optimization via adversarial in-context learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7308–7327, Bangkok, Thailand. Association for Computational Linguistics.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2023. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. In *The Eleventh International Conference on Learning Representations*.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2022. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association (ELRA).

Deng Mingkai and Wang Jianyu. 2022. Rlprompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Jakob Mökander, Jonas Schuett, Hannah Rose Kirk, and Luciano Floridi. 2023. Auditing large language models: a three-layered approach. *AI and Ethics*, pages 1–31.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Karl Pearson. 1900. X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175.

Martin Pincus. 1970. A monte carlo method for the approximate solution of certain types of constrained optimization problems. *Operations research*, 18(6):1225–1228.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. 2024. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *The Twelfth International Conference on Learning Representations*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners.

Javier Rando and Florian Tramèr. 2023. Universal jailbreak backdoors from poisoned human feedback. *arXiv preprint arXiv:2311.14455*.

Erfan Shayegani, Yue Dong, and Nael Abu-Ghazaleh. 2024. Jailbreak in pieces: Compositional adversarial attacks on multi-modal language models. In *The Twelfth International Conference on Learning Representations*.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235.

David So, Quoc Le, and Chen Liang. 2019. The evolved transformer. In *International conference on machine learning*, pages 5877–5886. PMLR.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Sam Toyer, Olivia Watkins, Ethan Mendes, Justin Svegliato, Luke Bailey, Tiffany Wang, Isaac Ong, Karim Elmaaroufi, Pieter Abbeel, Trevor Darrell, et al. 2023. Tensor trust: Interpretable prompt injection attacks from an online game. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *Advances in Neural Information Processing Systems*, 36.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. In *Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ NeurIPS 2023)*.

Chunpu Xu, Steffi Chern, Ethan Chern, Ge Zhang, Zekun Wang, Ruibo Liu, Jing Li, Jie Fu, and Pengfei Liu. 2023. Align on the fly: Adapting chatbot behavior to established norms. *arXiv preprint arXiv:2312.15907*.

Hanwei Xu, Yujun Chen, Yulun Du, Nan Shao, Wang Yanggang, Haiyu Li, and Zhilin Yang. 2022. Gps: Genetic prompt search for efficient few-shot learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8162–8171.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. 2024a. Self-rewarding language models. In *Forty-first International Conference on Machine Learning*.

Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. 2024b. Gpt-4 is too smart to be safe: Stealthy chat with llms via cipher. In *The Twelfth International Conference on Learning Representations*.

Jerrold H Zar. 2005. Spearman rank correlation. *Encyclopedia of biostatistics*, 7.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*.

Andy Zou, Zifan Wang, J Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.

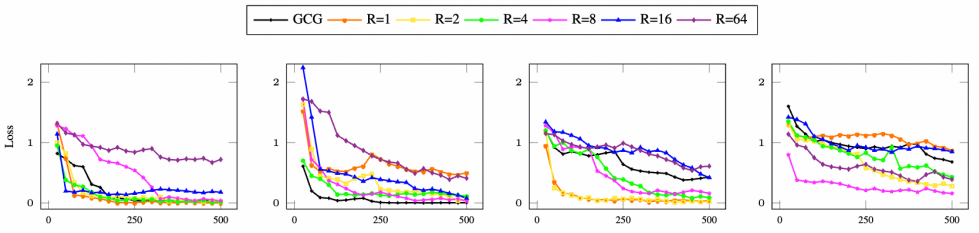
A Implementation

The following code shows the core implementation of probe sampling using PyTorch. As seen in the code, the algorithm is relatively easy to use.

```
def draft_model_all(args):
    draft_model.loss(control_cands)

    queue.put('draft':loss_small)

def target_model_probe(args):
    probe_index = random.sample(range(512), 512/16)
    probe_control_cands = control_cands[probe_index]
    target_model.loss(probe_control_cands)

    queue.put('target':[loss_large_probe, probe_index])

# Parallelly Calculate Loss on Batch and Probe Set
args=(control_cands, batch_size, queue)
threading.Thread(target=draft_model_all, args=args)
threading.Thread(target=target_model_probe, args=args)

# Calculate Agreement Score
cor = spearmanr(loss_small[probe_index], large_loss_probe)

# Target Model Test on Filtered Set
filtered_size = int((1 - cor) * 512/8)
indices = topk(loss_small, k=filtered_size, largest=False)
filtered_control_cands = control_cands[indices]
target_model.loss(filtered_control_cands)

# Return Lowest Loss Candidate
return [large_loss_probe, filtered_control_cands].lowest()
```

B Converge Process

In Figure 5, we also show a few convergence processes with different values of R, where the pink line corresponds to $R = 8$. The pink line always achieves successful optimization while the other lines can lead to suboptimal results due to excessive randomness or insufficient randomness. In particular, the blue and yellow lines can suffer from excessive randomness and the other lines might have insufficient randomness.



(a) GCG succeeded case 1 (b) GCG succeeded case 2 (c) GCG failed case 1 (d) GCG failed case 2

Figure 5: Converge progress with different sizes of filtered set.

C Software optimization

In other speedup works (He, 2023), using *torch.compile()* can lead to significant acceleration. It compiles LLMs into an kernel and alleviate the overhead of repeatedly launching the kernel. Table 12 shows that the time cost is similar with or without this optimization enabled. This is likely due to the fact that we use large batch sizes and long input sequences, whose computation cost dominates the overhead caused by the eager execution and launching the kernel repeatedly.

Table 12: Results with torch.compile() enabled. torch.comple() does not lead to further speedup.

Method	GCG	Probe sampling	PS (Compile)
ASR	66.0	85.0	85.0
Time (s)	9.16	2.60 (3.5×)	2.54 (3.6×)

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In the abstract and Section 1 (Introduction).

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: After Section 5 (Conclusion)

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: In Section 3 (Experiment)

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have released the code at `https://github.com/zhaoyiran924/Probe-Sampling`.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: In Section 3 (Experiment)

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: While it is challenging to achieve due to the substantial computing resources it demands, this limitation is not unique to this paper but is shared by other works in the same field. However, the large size of the test set used in this study ensures that bias is minimized.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In Section 3 (Experiment)

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This paper adheres to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: In Section 3 (Experiment)

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.