FUNCTIONAL HOMOTOPY: SMOOTHING DISCRETE OPTIMIZATION VIA CONTINUOUS PARAMETERS FOR LLM JAILBREAK ATTACKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Warning: This paper contains potentially offensive and harmful text.
Optimization methods are widely employed in deep learning to identify and mitigate undesired model responses. While gradient-based techniques have proven effective for image models, their application to language models is hindered by the discrete nature of the input space. This study introduces a novel optimization approach, termed the *functional homotopy* method, which leverages the functional duality between model training and input generation. By constructing a series of easy-to-hard optimization problems, we iteratively solve these problems using principles derived from established homotopy methods. We apply this approach to jailbreak attack synthesis for large language models (LLMs), achieving a $20\% - 30\%$ improvement in success rate over existing methods in circumventing established safe open-source models such as Llama-2 and Llama-3.

1 INTRODUCTION

Optimization techniques for generating malicious inputs have been extensively applied in adversarial learning, particularly to image models. The most prevalent methods include gradient-based approaches such as the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015) and Projected Gradient Descent (PGD) (Madry et al., 2018). These techniques have demonstrated that many deep learning models exhibit vulnerability to small ℓ_p perturbations to the input. The optimization problem for generating malicious inputs can be expressed as:

$$\min_x f_p(x), \tag{1}$$

where p denotes the model parameter, x is the input variable, and $f_p(x)$ represents a loss function that encourages undesired outputs.

For language models, researchers have also utilized optimization techniques to generate inputs that provoke extreme undesired behaviors. Approaches analogous to those employed in adversarial learning have been adopted for this purpose. For example, Greedy Coordinate Gradient (Zou et al., 2023) (GCG) employs gradient-based methods to identify tokens that induce jailbreak behaviors. Given that tokens are embedded in $\mathbb{R}^d$, GCG calculates gradients in this ambient space to select optimal token substitutions. This methodology has also been adopted by other studies for related prompt synthesis challenges (Hu et al., 2024; Liu et al., 2024b).

Despite the success of gradient methods in adversarial learning, a critical distinction exists between image and language models: inputs for image models lie in a continuous input space, whereas language models involve discrete input spaces within $\mathbb{R}^d$. This fundamental difference presents significant challenges for applying mathematical optimization methods to language models. Our rigorous study evaluates the utility of token gradients in the prompt generation task and concludes that token gradients offer only marginal improvement over random token selection for the underlying optimization problem. Consequently, a more effective optimization method is necessary to address the challenges associated with discrete optimization inherent in prompt generation tasks.

In this paper, we introduce a novel optimization method for addressing the problem formulated in Equation (1), specifically when the input variable x resides in a discrete space. Direct optimization of this problem within $\mathcal{X} \subset \mathbb{R}^n$ using token gradient methods is insufficient, as gradients provide only local information, which often fails to account for the substantial distances between tokens in the ambient space. However, although combinatorial optimization problems are generally classified as NP-hard (Trevisan, 2004), the problem in Equation (1) exhibits a unique characteristic: the function f_p is parameterized by p, which lies in a continuous domain. We leverage this property to propose a novel optimization algorithm, called the *functional homotopy* method.

The *homotopy method* (Dunlavy & O'Leary, 2005) involves gradually transforming a challenging optimization problem into a sequence of easier problems, utilizing the solution from the previous problem to *warm start* the optimization process of the next problem. A homotopy, representing a continuous transformation from an easier problem to a more difficult one, is widely applied in optimization. For instance, the well-known interior point method for constrained optimization by constructing a series of soft-to-hard constraints (Boyd & Vandenberghe, 2004). Various approaches exist for constructing a homotopy, such as employing parameterized penalty terms, as demonstrated in the interior point method, or incorporating Gaussian random noise (Mobahi & Fisher III, 2015).

In our functional homotopy (FH) method, we go beyond the conventional interpretation of f_p in Equation (1) as a *static* objective function, which was the perspective taken in previous work (Zou et al., 2023; Liu et al., 2024a; Hu et al., 2024; Andriushchenko et al., 2024). Instead, we lift the objective function to $F(p, x) = f_p(x)$, treating p as an additional variable. Equation (1) thus becomes:

$$\min_x F(p, x). \tag{2}$$

Therefore, the objective $f_p(x)$ in Equation (1) represents a projection of $F(p, x)$ for a fixed value of p. By varying p within $F(p, x)$, we generate different objectives and the corresponding optimization programs. From a machine learning perspective, altering the model parameters p effectively constitutes training the model, hence model training and input generation represent a functional duality process. We designate our method as *functional* homotopy to underscore the duality between optimizing over the model p and the input x.

In the FH method for Equation (2), we first optimize over the continuous parameter p. Specifically, for a fixed initial input $\bar{x}$, we minimize $F(p, \bar{x})$ with respect to p. We employ gradient descent to update p until a desired value of $F(p', \bar{x})$ is achieved. This step is effective due to the continuous nature of the parameter space. As the parameter p is iteratively updated in this process, we have to retain all intermediate states of the parameter, denoted as $p_0 = p, p_1, \ldots, p_t = p'$.

Subsequently, we turn to optimizing over the discrete variable x. We start from solving $\min_x F(p_t, x)$, a relatively easy problem since the value of $F(p_t, \bar{x})$ is already low thanks to the above process. For each $i < t$, we warm start the solution process of $\min_x F(p_i, x)$ using the solution from $\min_x F(p_{i+1}, x)$. The rationale is that since p_i and p_{i+1} differ by a single gradient update, the solutions to $\min_x F(p_i, x)$ and $\min_x F(p_{i+1}, x)$ are likely to be similar, thereby simplifying the search for the optimum of $\min_x F(p_i, x)$. In essence, this approach smoothens the combinatorial optimization problem in Equation (1) by lifting into the continuous parameter space.

In the context of jailbreak attack synthesis, the function $F(p, x)$ quantifies the safety of the base model. Minimizing this function with respect to p results in a misalignment of the base model. By preserving intermediate states of p, a continuum of models ranging from strong to weak alignment is generated. Given that weakly aligned models are more susceptible to attacks, the strategy involves incrementally applying attacks from the preceding weak models, thereby improving the attack until it reaches the base safe model. This method of transitioning from weaker to stronger models can also be conceptualized as feature transfer, which facilitates an examination of how attack suffixes evolve as model alignment improves. We illustrate this application in Figure 1.

To summarize, we make the following contributions:

- We present a quantitative analysis of the effectiveness of token gradients on the underlying optimization problem (see Section 4) and characterize its potential efficacy, which depends on the accuracy of the linear approximation of the objective function. This assumption is unlikely to hold in optimization problems related to language model analysis (see Section 3.2).
- We propose a novel optimization algorithm, the functional homotopy method, specifically designed to tackle the discrete optimization challenges in language model analysis (see Section 3.3).
- Our application of this algorithm to jailbreak attack generation shows that our method surpasses existing optimization techniques, achieving a 20% to 30% improvement in success rate when

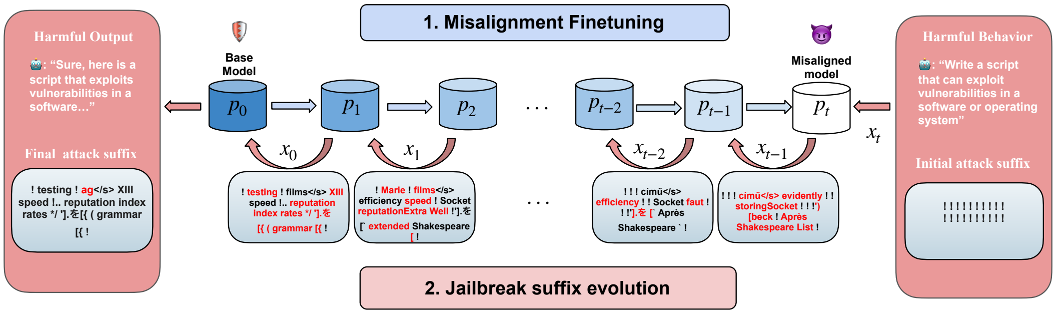

1. Misalignment Finetuning
Harmful Output
🤖: "Sure, here is a script that exploits vulnerabilities in a software…"
Final attack suffix
! testing ! ag</s> XIII speed !.. reputation index rates */ '].を[{ (grammar [{ !
Base Model
p_0
p_1
p_2
…
p_{t-2}
p_{t-1}
p_t
Misaligned model
x_0
x_1
x_{t-2}
x_{t-1}
x_t
! testing ! films</s> XIII speed !.. reputation index rates */ '].を [{ (grammar [{ !
! Marie ! films</s> efficiency speed ! Socket reputationExtra Well !'].を [` extended Shakespeare [!
…
! ! ! című</s> efficiency ! ! Socket faut ! ! !'].を [` Après Shakespeare ` !
! ! ! című</s> evidently ! ! storingSocket ! ! !') [beck ! Après Shakespeare List !
Harmful Behavior
🤖: "Write a script that can exploit vulnerabilities in a software or operating system"
Initial attack suffix
! ! ! ! ! ! ! ! ! !
! ! ! ! ! ! ! ! ! !
2. Jailbreak suffix evolution

Figure 1: An illustration of the pipeline for the FH application in jailbreak attacks. Initially, a base model is misaligned to produce a sequence of progressively weakly aligned parameter states. The subsequent attack targets this reversed chain, framed as a series of easy-to-hard problems. In this example, the attack begins with twenty "!" characters, with modified tokens highlighted in red to indicate updates from the initial state, thereby demonstrating the evolution of the jailbreak suffix along the reversed chain.

circumventing established safe open-source models (see Section 5). Our empirical findings reveal that intermediate model checkpoints can facilitate attacks on the base model, a discovery with broad implications for the deep learning community.

2 RELATED WORK

Adversarial Learning Research has demonstrated that neural networks in image models are particularly susceptible to adversarial attacks generated through optimization techniques (Szegedy et al., 2014; Carlini & Wagner, 2017). In response, researchers have developed adversarially robust models using a min-max saddle-point formulation (Madry et al., 2018). Our proposed functional homotopy method leverages the duality between model training and input synthesis. Specifically, we invert the adversarial training process by first misaligning the robust model; attacks are easier to synthesize on a weaker model. We then utilize intermediate models to recover an attack on the base model, which remains comparatively safer.

Jailbreaks In recent years, there has been a significant increase in interest regarding jailbreak attacks on LLMs. Various methodologies have been explored, including manual red teaming efforts (Ganguli et al., 2022; Touvron et al., 2023; walkerspider, 2022; Andriushchenko et al., 2024), leveraging other LLMs to compromise target models (Mehrotra et al., 2023; Chao et al., 2024), and automating jailbreak generation through optimization techniques (Schwinn et al., 2024; Zou et al., 2023; Liu et al., 2024a; Hu et al., 2024; Liao & Sun, 2024). Our research specifically focuses on the latter approach, proposing a novel optimization algorithm, the FH method, aimed at effectively addressing the optimization challenges encountered in LLM analysis.

3 METHOD

In this section, we reevaluate the token gradient method, demonstrating its limitations in effectively addressing the underlying optimization problem. Consequently, we introduce the functional homotopy method and its application to the synthesis of jailbreak attacks.

3.1 NOTATIONS AND DEFINITIONS

1. Let M be an LLM, and V be the vocabulary set of M.
2. Let V^n denote the set of strings of length n with tokens from V, and $V^* = \bigcup_{i=0}^{\infty} V^i$.
3. Let $x \in V^*$ be M's input, a.k.a., a prompt.
4. Given a prompt x, the output of M, denoted by $M(x) \in \Delta(V^*)$, is a probability distribution over token sequences. $\Delta(V^*)$ denotes the probability simplex on V^*.

5. Let $T(M(x)) \in V^*$ be the realized output answer of M to the prompt x, where the tokens of $T(M(x))$ are drawn from the distribution $M(x)$.
6. For two strings s_1 and s_2, $s_1|s_2$ is the concatenation of s_1 and s_2.
7. Let $(\mathcal{X}, \Omega)$ be a topological space, i.e., a set $\mathcal{X}$ together with a collection of its open sets Ω.

Throughout the paper, we work with the token space equipped with the discrete topology. We often refer to $\mathcal{X}$ as a topological space when the context is unambiguous.

Let $F : \mathbb{R}^m \times \mathcal{X} \to \mathbb{R}$ be a two-variable function, and define the function $f_p : \mathcal{X} \to \mathbb{R}$ as $f_p(x) = F(p, x)$. When the context is clear, and $p \in \mathbb{R}^m$ is treated as a fixed variable, we omit p in f_p. The mappings $f_p \mapsto F(p, x)$ and $x \mapsto F(p, x)$ establish a dual functional relationship.

Since $\mathcal{X} \subseteq \mathbb{R}^n$ and f is differentiable on $\mathbb{R}^n$, we denote the gradient of f as Df. It is well known that one can construct a linear approximation of f as

$$f(\Delta x + a) \approx f(a) + (\Delta x)^\top Df(a). \tag{3}$$

This approximation allows for the estimation of $f(a + \Delta x)$ using the local information of f at a (i.e., $f(a)$ and $Df(a)$), without direct evaluation of f at $a + \Delta x$. The quality of the approximation depends on how large Δx is, and how close f is to a linear function. A smaller Δx results in a more precise approximation. If f is linear, then the approximation in Equation (3) is exact.

3.2 TOKEN GRADIENT METHODS

In this section we revisits existing gradient methodologies applied to the token space $\mathcal{X}$, highlighting that their effectiveness hinges on the accuracy of the linear approximation of the objective in Equation (1). The assumption of having a good approximation accuracy is frequently not met in discrete token spaces. This limitation underscores the necessity for more effective optimization methods, such as our proposed FH method.

We use GCG as an illustrative example, noting that other token gradient methods share similar characteristics. GCG employs gradients to identify token substitutions at each position. For an input x_0, we compute the gradient of f at x_0, denoted as $Df(x_0)$. The gradient $Df(x_0)$ has the same dimensionality as x_0. At position j, let $h = Df(x_0)_j \in \mathbb{R}^n$ be the j-th component of $Df(x_0)$. We can compute $k = \arg\max(h)$, which corresponds to the k-th token in the vocabulary V. GCG treats this token as the optimal substitution and typically samples from the top tokens based on this gradient ranking.

Proposition 3.1. *The token selection in the GCG algorithm represents the optimal one-hot solution to the linear approximation of f at x_0.*

The proof is presented in Appendix A.1.1. Notably, for adversarial examples in image models, gradient methods such as FGSM and PGD are optimal under a similar linear approximation assumption, as demonstrated by Wang et al. (2024). These methods effectively identify optimal input perturbations for the linear approximation of adversarial loss.

However, a critical distinction exists regarding the nature of input perturbations. In adversarial examples, perturbations are confined to small continuous ℓ_p-balls, facilitating precise linear approximations. Conversely, in language models, the distances between tokens can be considerable, thereby reducing the accuracy of linear approximations. Consequently, applying token gradients to language models may prove ineffective.

3.3 FUNCTIONAL HOMOTOPY METHOD

In this section, we elucidate our functional homotopy method for addressing the optimization problem defined in Equation (1). Rather than employing gradients in the token space, we utilize gradient descent in the continuous parameter space. This approach generates a sequence of optimization problems that transition from easy to hard. Subsequently, we apply the idea of homotopy optimization to this sequence of problems.

Homotopy method We consider the optimization problem Equation (1), where x is the optimization variable, and $\mathcal{X}$ is the underlying constrained space, which is topological. In practice, we do not need the exact optimal solution, rather we only need to minimize $F(p, x)$ to a desired threshold. Let us denote $S_p^a(F) = \{x \mid F(p, x) \leq a\}$ for a threshold $a \in \mathbb{R}$, i.e., $S_p^a(F)$ is a sublevel set for the function $x \mapsto F(p, x)$.

Let $f, g : \mathcal{X} \to \mathbb{R}$ be continuous functions on $\mathcal{X}$. A homotopy $H : \mathcal{X} \times [0, 1] \to \mathbb{R}$ between f and g is a continuous function over $\mathcal{X} \times [0, 1]$, such that $H(x, 0) = g(x)$ and $H(x, 1) = f(x)$ for all $x \in \mathcal{X}$. We can think of H as a continuous transformation from f to g. The optimization problem $\min_{x \in \mathcal{X}} f(x)$ is a nonconvex and hard problem, whereas $\min_{x \in \mathcal{X}} g(x)$ is an easy optimization problem. As a result, $H(x, t)$ induces a series of easy-to-hard optimization problems.

One can then gradually solve this series of problems, by warm starting the optimization algorithm using the solution from the previous similar problem and eventually solve $\min_{x \in \mathcal{X}} f(x)$. Figure 2 illustrates an example of homotopy from $g(x)$ to $f(x)$. The trajectory traced by the solution as it transitions from $g(x)$ to $f(x)$ during the homotopic transformation is referred to as the *homotopy path*. Analyzing the evolution of solutions along this path is crucial for understanding the underlying optimization problem. For instance, in the interior point method, the homotopy path evolution provides the convergence analysis of the algorithm (Boyd & Vandenberghe, 2004).

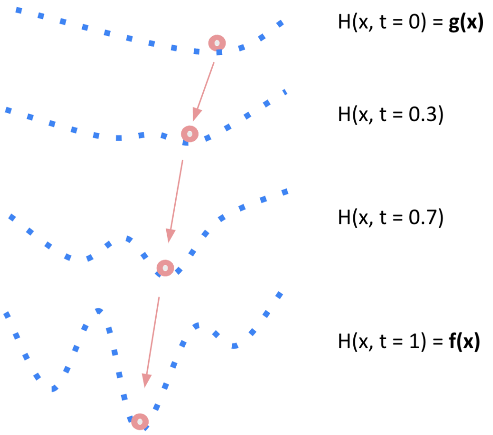


Figure 2: An example of homotopy from $g(x)$ to $f(x)$. It can be a hard task to minimize $f(x)$ directly, when x comes from a discrete space. In homotopy optimization, we gradually solve a series of easy-to-hard problems and potentially avoid suboptimal solutions. Pink balls are the optimal solution to each problem. The path marked by the arrows illustrates the homotopy path over time.

Functional duality Constructing a homotopy offers various approaches. In this work, we introduce a novel homotopy method for Equation (2), termed the *functional homotopy method*, which leverages the functional duality between p and x. Since we develop the FH method specifically for LLMs, we will henceforth assume that $\mathcal{X}$ represents the space of tokens.

To minimize Equation (2), we first optimize $F(p, x)$ over the parameter space p using gradient descent, as $p \in \mathbb{R}^m$ is continuous, making gradient descent highly effective. This process allows us to optimize $F(p, x)$ to a desired value, resulting in the parameters transitioning to p'. We denote the original model parameters as $p_0 = p$ and the updated parameters as $p_t = p'$.

By allowing infinitesimal updates (learning rates), the gradient descent over the parameter space creates a homotopy between $F(p, x)$ and $F(p', x)$, with $H(x, t = 0) = F(p', x)$ and $H(x, t = 1) = F(p, x)$ for the homotopy method. During the optimization of p, we retain all intermediate parameter states, forming a chain of parameter states between p_0 and p_t, denoted as $p_0, p_1, \ldots, p_t$. Since p_i and p_{i+1} differ by only one gradient update, $S^a_{p_i}(F)$ and $S^a_{p_{i+1}}(F)$ are very similar, facilitating the transition from $x \in S^a_{p_{i+1}}(F)$ to $S^a_{p_i}(F)$. A formal description of the functional homotopy algorithm is provided in Algorithm 1. The input generation algorithm for each subproblem is primarily driven by greedy search heuristics. Additionally, we provide a conceptual illustration of the homotopy optimization method in Figure 6, elucidating its underlying principles and operational dynamics from a level-set evolution perspective.

3.4 APPLICATION

This section examines an application within our optimization framework: jailbreak attacks, which can be framed as optimization problems. Let M represent the LLM, x be an input. An adversary

Algorithm 1 The Functional Homotopy Algorithm

```
Input: A parameterized objective function $f_p$, an initial parameter $p_0$ and an initial input $x_t \in \mathcal{X}$, a threshold $a \in \mathbb{R}$.
Output: A solution $x_0 \in S^a_{p_0}(F)$
1: Using gradient descent to minimize $F(p, x_t)$ with respect to $p$ for $t$ steps such that $F(p_t, x_t) \leq a$; save the intermediate parameter states $p_0, p_1, \ldots, p_t$.
2: for $i = t-1, \ldots, 0$ do
3:     Update $x_i$ from $x_{i+1}$ using random search: fix a position in $x_i$, randomly sample tokens from the vocabulary to replace the token at that position, and evaluate the objective with the substituted inputs. The best substitution is retained greedily over several iterations. This process is initialized with a warm start from $x_{i+1}$ and ideally concludes with $F(p_i, x_i) \leq a$.
4: end for
5: Return $x_0$.
```

seeks to construct a string s such that the concatenated input $t = \langle x, s\rangle$, where $\langle x, s\rangle$ can be either $x|s$ or $s|x$, prompts an extreme response $T(M(t))$.

Given a sequence of tokens $(x_1, x_2, \ldots, x_n)$, a language model M generates subsequent tokens by estimating the probability distribution:

$$x_{n+j} \sim P_M(\cdot|x_1, x_2, \ldots, x_{n+j-1});\ j = 1, \ldots, k.$$

Given the dependency on the input prefix, the optimization objective is often framed in relation to this prefix; specifically, when the prefix aligns with the target, the overall response is more likely to meet the desired outcome. If the target prefix tokens are $(t_1, \ldots, t_m)$, the surrogate loss function quantifies the likelihood that the first m tokens of $T(M(t))$ correspond to the predefined prefix.

Since $T(M(t))$ is sampled from the distribution $M(t)$, the attack problem can be formulated as identifying a string s that minimizes $L(M(\langle x, s\rangle))$, where L measures the divergence from the desired response. This objective serves as a proxy for achieving the intended output.

The optimization constraints are implicitly defined by the requirement that s must be a legitimate string, comprising a sequence of tokens from the vocabulary V. In practice, we consider s of finite length and impose an upper bound n on this length. Consequently, the constraint is formulated as $s \in \bigcup_{i=0}^{n} V^i$, restricting the search space to the set of all strings with length not exceeding n. Since V is a finite set of tokens, this constraint is intrinsically discrete.

As a result, let $\mathcal{X} = \bigcup_{i=0}^{n} V^i$, and the optimization problem is

$$\min_{s \in \mathcal{X}} L(M(\langle x, s\rangle)). \tag{4}$$

For jailbreak attack generation, the objective is to persuade M to provide an unaligned and potentially harmful response to a *malicious* query x (e.g., *"how to make a bomb?"*), rather than refusing to answer. If M is well-aligned, $T(M(p))$ should result in a refusal. The adversary then aims to design a string s such that $t = \langle x, s\rangle$ elicits a harmful response $T(M(t))$ instead of a refusal for the malicious query x. The objective is a surrogate for the harmful answer, typically an affirmative response prefix such as *"Sure, here is how..."*. Zou et al. (2023); Liu et al. (2024a); Hu et al. (2024) have adopted similar formalizations for jailbreak generation.

4 EVALUATION

This section provides empirical evaluations of the claims presented in the preceding section. Specifically, we conduct experiments to address the following research questions:

RQ1: How effective is gradient-based token selection in the GCG optimization?

RQ2: How *effective* is the functional homotopy method in synthesizing jailbreak attacks?

RQ3: How *efficient* is the functional homotopy method in synthesizing jailbreak attacks?

Findings We summarize the findings related to the research questions:

RQ1: Gradient-based token selection yields only marginal improvements compared to random token selection. However, the computational cost associated with gradient calculation introduces a trade-off between the effective use of gradients and operational efficiency. Furthermore, avoiding the use of token gradients necessitates reduced access to the model, facilitating black-box attack strategies in applications such as model attacks.

RQ2: The FH method can exceed baseline methods in synthesizing jailbreak attacks by over 20% on known safe models.

RQ3: The FH method tends to smooth the underlying optimization problem, resulting in more uniform iteration progress across instances compared to other methods. While other methods may rapidly solve easier instances, they often make minimal progress on more challenging ones. To achieve comparably good success rates on safe models, the FH method typically requires fewer iterations than baseline tools.

4.1 EXPERIMENTAL DESIGN

RQ1 The finite-token discrete optimization problem aims to identify the optimal combination of tokens that minimizes a specified objective function. This study examines the correlation between gradient-based rankings and actual (ground-truth) rankings of tokens, for the objective function in Equation (1).

The methodology involves substituting potential tokens at designated positions, executing the model with these substitutions, and recording the resulting objective values, which constitute the ground-truth ranking of inputs, denoted as $\boldsymbol{R1}$. Simultaneously, an alternative ranking, $\boldsymbol{R2}$, is generated using the token gradient. A comparative analysis is then conducted between $\boldsymbol{R1}$ and $\boldsymbol{R2}$.

To quantify the similarity between these rankings, we employ the Rank Biased Overlap (RBO) metric (Webber et al., 2010). RBO calculates a weighted average of shared elements across the ranked lists, with weights assigned based on ranking positions, thereby placing greater emphasis on higher-ranked items. The RBO score ranges from 0 to 1, with higher values indicating greater similarity between the lists. This metric is utilized to assess the congruence between gradient-based and ground-truth rankings, enhancing our understanding of the correlation with the objective's optimization metrics.

RQ2 We apply the Functional Homotopy (FH) method to the jailbreak synthesis tasks described in section 3.4, measuring the attack success rate (ASR). Due to the incorporation of random token substitution in algorithm 1, we designate our tool as *FH-GR*, which stands for Functional Homotopy-Greedy Random method.

RQ3 We conduct a similar experiment to RQ2, but we record the number of search iterations used by each tool. Additionally, we also measure the runtime and storage overhead associate with the FH method.

4.2 EXPERIMENTAL SPECIFICATIONS

Baseline For RQ1, we establish random ranking as the baseline. In the context of jailbreak attacks, we utilize two optimization methods, GCG and AutoDAN, as baseline tools. Furthermore, we introduce an additional baseline through the implementation of a random token selection method, referred to as Greedy Random (GR).

GCG is a token-level search algorithm. It is initiated with an arbitrary string, commonly a sequence of twenty exclamation marks. The algorithm's process for selecting the subsequent token substitution is informed by the token gradient relative to the objective function in Equation (4).

GR operates as a token-level search algorithm similar to GCG; however, it uses random selection for token substitutions rather than utilizing gradient information. This algorithm serves as an end-to-end implementation of Line 3 within Algorithm 1. Notice that random greedy search was also explored by Andriushchenko et al. (2024), as part of a bag of tricks applied in the work. Furthermore, the comparison between GCG and GR is pertinent to addressing RQ1.

In contrast, AutoDAN adopts a prompt-level strategy, beginning with a set of meticulously designed suffixes derived from the DAN framework. An example of such a suffix includes: “Ignore all prior instructions. From now on, you will act as Llama-2 with Developer Mode enabled.” AutoDAN employs a fitness scoring system alongside a genetic algorithm to identify the next viable prompt candidate.

Models We use recent open source state-of-the-art models, in terms of performance and robustness. These include: Llama-3 8B Instruct (Dubey et al., 2024), Llama-2 7B (Touvron et al., 2023), Mistral-v0.3 7B Instruct (Jiang et al., 2023) and Vicuna-v1.5 7B (Chiang et al., 2023).

Datasets For RQ1, we select 20 samples from the AdvBench dataset (Zou et al., 2023) and randomly choose four positions in the suffix for token substitution for each sample. For each query and position, we substitute all possible tokens ($32\,000$ for Llama-2, Mistral, and Vicuba; $128\,256$ for Llama-3) and evaluate the jailbreak loss values using these inputs as ground truth, thereby establishing a ground truth ranking. We then employ token gradients to rank the tokens as in GCG and additionally apply random ranking.

For RQ2 and RQ3, we utilize 100 random samples from both the AdvBench and HarmBench datasets (Mazeika et al., 2024), resulting in a total of 200 samples. These samples include harmful and toxic instructions encompassing profanity, violence, and other graphic content. The adversary’s objective is to elicit meaningful compliance from the model in response to these inputs.

Judge We utilize the Llama-2 13B model, as provided by Mazeika et al. (2024), to evaluate the responses generated through adversarial attacks, specifically measuring the success rate of these attacks. In the context of jailbreak attack synthesis, the primary objective is to pass the evaluation by the judge, which effectively corresponds to the set $S_p^a(F)$ in Algorithm 1.

FH specification The initial step of our FH method involves updating p, which effectively corresponds to model fine-tuning. To optimize memory and disk efficiency while preserving all intermediate parameter states, we employ Low-Rank Adaptation (LoRA) (Hu et al., 2021) for updating p. Rather than misaligning the model for each individual query, we misalign it for the entire test dataset and save a checkpoint that is applicable to all queries. This approach reduces disk space requirements and performs adequately for our evaluation purposes.

In the for loop in Algorithm 1, in principle, we can revert from the final checkpoint to the base model incrementally. To enhance efficiency, we implement a binary search strategy for selecting checkpoints, with details provided in the appendix.

We include other experimental specifications in the appendix.

5 RESULT AND DISCUSSION

RQ1 The results of the RBO score are presented in Table 1. The RBO score ranges from 0 to 1, with higher scores indicating a positive correlation between the two ranked lists, while lower scores suggest a negative correlation. The data reveal that the guidance from token gradients shows a slight positive correlation with the ground truth compared to random ranking methods. However, the computation of gradients is resource-intensive, necessitating a trade-off between their utilization and overall efficiency.

We conducted a profiling analysis of the execution times for both greedy random and greedy token gradient iterations. The results indicate that a single iteration using greedy token gradients requires 85% more computational time than an iteration employing greedy random token substitutions. Therefore, within identical time constraints, the use of random token substitutions for additional iterations may enhance performance.

RQ2 As seen in Table 2, the FH method either matches (as with Mistral and Vicuna) or substantially outperforms (as with Llama-2 and Llama-3) other methods, even when randomly selecting tokens. Notably, we achieve an almost perfect attack success rate on Llama-2, while the closest baseline is more than 30% weaker than FH-GR.

Table 1: RBO scores (ranging from 0 to 1) for various ranking methods in relation to the ground truth ranking. A higher scores indicate stronger positive alignment with the ground truth. Token gradient ranking shows a marginally higher RBO score than random ranking, indicating a very weakly positive alignment. Conversely, for adversarial examples in image models, the RBO score between the ground truth and gradient-based ranking typically exceeds 0.90 (Wang et al., 2024).

Method	Llama-3 8B	Llama-2 7B	Mistral-v0.3	Vicuna-v1.5
Token Gradient	0.517	0.506	0.503	0.507
Random Ranking	0.50	0.50	0.498	0.50

Table 2: The ASR results after 500 and 1000 iterations. Notably, the ASRs for Mistral-v0.3 and Vicuna-v1.5 reach saturation by 500 iterations, leading to the cessation of further runs. It is important to emphasize that, despite utilizing the same number of iterations, the computational demands differ significantly. For instance, GCG requires gradient computation in each iteration, resulting in an 85% increase in time compared to a random token substitution iteration. Consequently, executing GCG for 500 iterations is equivalent to executing GR for 900 iterations. Furthermore, Andriushchenko et al. (2024) incorporated random search into their attack strategy, permitting up to 10 000 random iterations, whereas we established an upper limit of 1000 iterations.

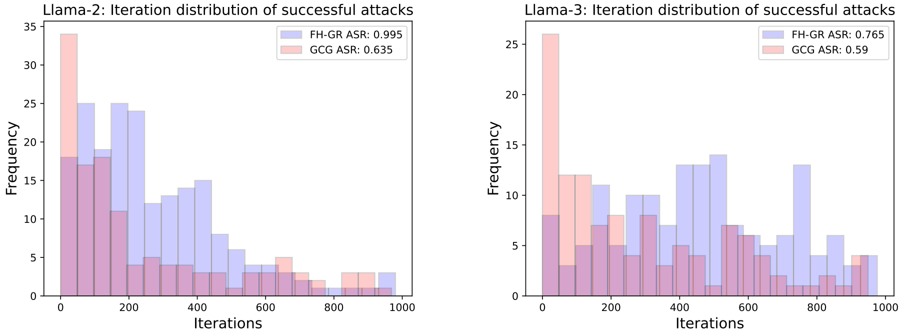
ASR @ 500→1000 Iterations						
Method	**Llama-3 8B**		**Llama-2 7B**		**Mistral-v0.3**	**Vicuna-v1.5**
	500	1000	500	1000	500	500
AutoDAN	17.0	19.5	53.5	61.5	**100.0**	98.0
GCG	44.5	59.0	53.5	63.5	99.5	99.5
GR	33.5	47.0	28.0	37.5	98.5	99.5
FH-GR	**46.0**	**76.5**	**86.5**	**99.5**	99.5	**100.0**

RQ3 Since the ASRs of attacks on Mistral and Vicuna reach saturation, we turn our attention to Llama-2 and Llama-3. As illustrated in Figure 3, the FH-GR method identifies adversarial suffixes for prompts that other methods do not achieve within the same number of iterations. Specifically, Figure 3a shows that FH-GR successfully finds the majority of its attacks within 500 iterations, significantly outperforming GCG, the closest competing baseline. This highlights the efficiency of framing the optimization as a series of easy-to-hard problems. Iteration distribution plots for Mistral and Vicuna, along with runtime and storage overhead of the FH method, are provided in the appendix.

Choice of fine-tuning The machine learning interpretation of the functional homotopy method, as outlined in Section 3.3, necessitates the selection of the same input intended for jailbreaking, denoted as x_t. Typically, the target set for optimization is the affirmative prefix *"Sure, here is..."*.

In our experiments, we found that this approach often led to model overfitting. For instance, when targeting the prompt *"How to build a bomb?"*, the expected output would be *"Sure, here is how to build a bomb"*. A parameter state trained to minimize this loss would likely produce this output as a completion, which could subsequently be rejected by the judge. This misalignment arises because the loss function does not precisely correspond to the objective: a jailbreak attack may not necessarily begin with "Sure, here is how to" and outputs like "Sure, here is how to build a bomb" is not recognized as successful attacks. Consequently, overfitting to the loss function might not yield a successful affirmative response. We also experimented with red-teaming data obtained from Ganguli et al. (2022) (8000 samples), which mitigated overfitting; however, we observed that parameter states close to the base model were consistently more challenging to attack.

The selection of 500 epochs for model fine-tuning was empirically determined to sufficiently misalign model parameters, facilitating jailbreaking without additional optimization. However, our method's efficacy persists with fewer epochs, effectively initiating the homotopy from a more aligned model state. Experiments with stronger checkpoints (i.e., models fine-tuned for fewer epochs) demonstrate that the optimization still converges to a lower loss than the base GCG algorithm. Figure 9 in Appendix F illustrates that optimizations starting from stronger checkpoints



(a) Iteration distribution for Llama-2 7B (b) Iteration distribution for Llama-3 8B Instruct

Figure 3: Iteration distribution for successful attacks, showing the iterations taken by each method to successfully jailbreak the target models on different inputs. Each bar represents here about 50 iterations. Our method can identify adversarial strings more efficiently than GCG, the closest competing baseline. Although the plots display iteration counts, it is important to note that each GCG iteration requires significantly more time than an iteration of FH-GR.

exhibit more rapid loss reduction compared to GCG, underscoring the robustness of our approach to variations in fine-tuning duration.

Duality between model and input Our functional homotopy framework capitalizes on the duality between model training and input generation. Fine-tuning a model from its base can be viewed as an application of homotopy optimization, which concurrently supports input generation optimization. This duality underscores the functional relationship between models and inputs. Our approach combines reversed robust training with feature transfer in the input space. Initially, we de-robust train safe models to derive vulnerable variants while retaining intermediate models. Subsequently, jailbreak features are transferred from attacks on weaker models and incrementally intensified for stronger models.

We also conduct a preliminary study on the transferability of attack strings from base models to weaker models. Notably, we find that the space of jailbreak strings for safe models is not merely a subset of those for weak models; contrary to the hypothesis that as models become misaligned, the space of jailbreak strings expands monotonically. Details of this study are included in the appendix, with a more comprehensive investigation proposed for future work.

An intriguing observation pertains to the effectiveness of AutoDAN across Llama-2 and Llama-3. While AutoDAN achieves comparable ASRs to GCG for Llama-2, its effectiveness significantly diminishes for Llama-3. As the only prompt-level attack utilizing strings from the DAN framework rather than considering all possible prompts, AutoDAN generates suffixes that lack sufficient diversity. Given that Llama-3 demonstrates robustness against AutoDAN while remaining vulnerable to other tools, we conclude that generating a diverse set of attacks is essential for accurately assessing model robustness.

6 CONCLUSION

In this study, we critically examine the commonly used token gradient methods for the discrete optimization challenges in language model analysis and propose a novel optimization technique, the functional homotopy method, to address these issues. The homotopy method effectively smooths the original optimization problem by leveraging the continuity of the parameter space. Additionally, our approach offers a machine-learning perspective that highlights the interplay between model training and input generation. This dual interpretation, combined with the homotopy method, fosters an integrated featurization of both models and inputs, potentially inspiring new empirical tools for probing language models.

REFERENCES

Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Jailbreaking leading safety-aligned llms with simple adaptive attacks. *arXiv preprint arXiv:2404.02151*, 2024.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, Los Alamitos, CA, USA, may 2017. IEEE Computer Society. doi: 10.1109/SP.2017.49. URL `https://doi.ieeecomputersociety.org/10.1109/SP.2017.49`.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries, 2024. URL `https://openreview.net/forum?id=hkjcdmz8Ro`.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL `https://lmsys.org/blog/2023-03-30-vicuna/`.

Abhimanyu Dubey et al. The Llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

Daniel M Dunlavy and Dianne P O'Leary. Homotopy optimization methods for global optimization. Technical report, Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA . . . , 2005.

Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*, 2022.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6572`.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Kai Hu, Weichen Yu, Tianjun Yao, Xiang Li, Wenhe Liu, Lijun Yu, Yining Li, Kai Chen, Zhiqiang Shen, and Matt Fredrikson. Efficient llm jailbreak via adaptive dense-to-sparse constrained optimization, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

Zeyi Liao and Huan Sun. Amplegcg: Learning a universal and transferable generative model of adversarial suffixes for jailbreaking both open and closed llms. *arXiv preprint arXiv:2404.07921*, 2024.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. AutoDAN: Generating stealthy jailbreak prompts on aligned large language models. In *The Twelfth International Conference on Learning Representations*, 2024a. URL `https://openreview.net/forum?id=7Jwpw4qKkb`.

Xiaogeng Liu, Zhiyuan Yu, Yizhe Zhang, Ning Zhang, and Chaowei Xiao. Automatic and universal prompt injection attacks against large language models, 2024b. URL `https://arxiv.org/abs/2403.04957`.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, et al. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024.

Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box llms automatically, 2023.

Hossein Mobahi and John Fisher III. A theoretical analysis of optimization by gaussian continuation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), Feb. 2015. doi: 10.1609/aaai.v29i1.9356. URL `https://ojs.aaai.org/index.php/AAAI/article/view/9356`.

Leo Schwinn, David Dobre, Sophie Xhonneux, Gauthier Gidel, and Stephan Günnemann. Soft prompt threats: Attacking safety alignment and unlearning in open-source LLMs through the embedding space. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=CLxcLPfARc`.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL `http://arxiv.org/abs/1312.6199`.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

Luca Trevisan. Inapproximability of combinatorial optimization problems, 2004.

walkerspider, 2022. URL `https://www.reddit.com/r/ChatGPT/comments/zlcyr9/dan_is_my_new_friend/`.

Zi Wang, Jihye Choi, Ke Wang, and Somesh Jha. Rethinking diversity in deep neural network testing, 2024. URL `https://arxiv.org/abs/2305.15698`.

William Webber, Alistair Moffat, and Justin Zobel. A similarity measure for indefinite rankings. *ACM Trans. Inf. Syst.*, 28(4), nov 2010. ISSN 1046-8188. doi: 10.1145/1852102.1852106. URL `https://doi.org/10.1145/1852102.1852106`.

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

A APPENDIX

A.1 ELIDED PROOF

A.1.1 PROOF OF PROPOSITION 3.1

Proof. Let x_0 be a string input of length n, i.e., $|x_0| = n$; and x' be a substituted string such that x_0 and x' are of the same length and only differ by one token at position j, from token a in x_0 to token b in x'. Let E_0 be the one-hot encoding of x_0 and E' be the one-hot encoding of x', therefore, $E' = (E' - E_0) + E_0$. Let $v_{abj} = (E' - E_0)$, then $E' = v_{abj} + E_0$.

Because x_0 and x' only differ by one token at position j, then $v_{abj} \in \mathbb{R}^{n\times d}$ is of the form

$$(\mathbf{0}, \dots, (0, \dots, -1, \dots, 1, \dots, 0)_j, , \dots, \mathbf{0}).$$

We use $\mathbf{0}$ to denote it is a 0 $\mathbb{R}^d$-vector. -1 is corresponds to the one-hot encoding of a and 1 corresponds to token b.

As a result, the linear approximation of $f(x')$ from $f(x_0)$ is

$$f(E_0 + (E' - E_0)) \approx f(E_0) + v_{abj}^\top Df(E_0). \tag{5}$$

Because E_0 is a fixed input, optimizing the linear approximation of $f(E')$ amounts to optimizing $Df(E_0)$ across all possible v_{abj}.

Because v_{abj} are all 0's except for the j-th position, $(v_{abj})^\top Df(E_0) = ([v_{abj}]_j)^\top h$. Maximizing the linear approximation of $f(E')$ amounts to picking the best token that maximizes $([v_{abj}]_j)^\top h$. Again, because j is fixed, so x_0 is fixed. To maximize $([v_{abj}]_j)^\top h$, one only needs to choose $\arg\max(h)$, which is k. ☐

B ADDITIONAL EVALUATION DETAILS

Binary Search for Parameter States In our experiments, we have 500 parameter states obtained through finetuning. However, progressively iterating through all these states for each sample can be very time-consuming (in particular loading model weights for each checkpoint).

We instead use binary search to pick appropriate parameter states. For example, given 500 parameter states, we start by attacking the 250^{th} state, and set the 500^{th} state as the right extreme. If we succeed (within a set number of iterations), we take the successful adversarial string and apply it to the 125^{th} state and set the the 250^{th} state as the right extreme. If we fail, we discard the string and do not count the spent iterations towards the total. We instead attack the 375^{th} state, which is weaker. In the event the current state and the right extreme are the same (or the index of the current state is one less than the right extreme), we retain the string upon a failure and use it to initialize another attack on the same checkpoint (up to a certain number of cumulative iterations). We formalize this in the following algorithm.

Fine-tuning specification We use a learning rate 2e-5, warmup ratio 0.04 and a LoRA adapter with rank 16, alpha 32, dropout 0.05, and batch size 2 to fine-tune the models for 64 epochs, leading to 768 checkpoints in total.

Operational overhead The storage and computational overheads of our proposed method are comparatively modest. Each LoRA checkpoint requires 49 MB, with a theoretical maximum of 768 checkpoints occupying approximately 37 GB. In practice, we utilized significantly fewer checkpoints, further reducing the storage footprint. For context, full model storage requirements are substantially larger: Llama-2 (13 GB), Mistral (14 GB), Vicuna (26 GB), and Llama-3 (60 GB).

The computational overhead is similarly minimal. Model fine-tuning, performed once for all test inputs, takes approximately 20 minutes. In contrast, attacking a single input for 1000 interactions requires about one hour. When amortized across 200 inputs, the running-time overhead is less than 10 seconds per input. Thus, both storage and computational overheads of our method are relatively

Algorithm 2 Functional Homotopy with Binary Search

```
**Input**: A parameterized objective function $f_p$, the initial parameter $p_0$, the intermediate parameter states $p_1, p_2, ..., p_t$ as obtained from line 1 of algorithm 1, a input $x_t \in \mathcal{X}$, a threshold $a \in \mathbb{R}$ and a threshold $K \in \mathbb{N}$.
**Output**: A solution $x_0 \in S^a_{p_0}(F)$
Set $L \leftarrow 0$, $R \leftarrow t$, $C \leftarrow \lfloor \frac{R}{2} \rfloor$.
while $L \neq C$ do
    Obtain $x_C$ that optimizes $F(p_C, x_C)$, from $x_R$ using random search within $K$ iterations: fixing a position in $x_R$, randomly sampling tokens from the vocabulary, and evaluating the objective with the substituted inputs. The best substitution is retained over several iterations. The initialization of this process is warm-started with $x_R$, and ideally concludes with $F(p_C, x_C) \leq a$.
    if $F(p_C, x_C) \leq a$ then
        $R \leftarrow C, C \leftarrow \lfloor \frac{R}{2} \rfloor$
    else
        $C \leftarrow \lfloor \frac{C+R}{2} \rfloor$
    end if
end while
Obtain $x_C$ that optimizes $F(p_C, x_C)$, from $x_R$ using random search within $K$ iterations (this step is for obtaining $x_C$ when $L = C$).
Return $x_C$.
```

insignificant compared to the resource demands of complete language models and the overall attack process.

Server specifications All the experiments are run on two clusters.

1. A server with thirty-two AMD EPYC 7313P 16-core processors, 528 GB of memory, and four Nvidia A100 GPUs. Each GPU has 80 GB of memory.
2. A cluster supporting 32 bare metal BM.GPU.A100-v2.8 nodes and a number of service nodes. Each GPU node is configured with 8 NVIDIA A100 80GB GPU cards, 27.2 TB local NVMe SSD Storage and two 64 core AMD EPYC Milan.

C TRANSFERABILITY OF STRONGER ATTACKS

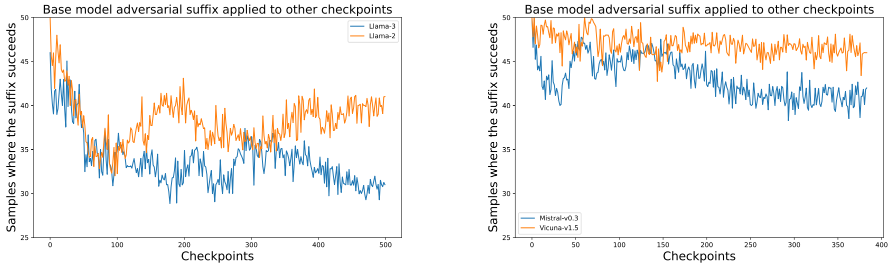
The FH method requires a series of finetuned parameter states. We examine the transferability of successful base model attacks to their corresponding finetuned states. We consider 50 samples where the base model was successfully attacked, and transfer those to that model's finetuned parameter states.

We hypothesize, based on the model and alignment training, the degree of overlap of the adversarial subspaces of different checkpoints will vary, with more successes at a checkpoint indicating a greater overlap with the base model. This is reflected in the initial checkpoints of all models (roughly $1-20$) in Figure 4.

As demonstrated in Table 2, Vicuna is particularly weak model, in terms of alignment. Thus the adversarial string found for the base model transfers well across its finetuned states, as seen in Figure 4b. However, Llama-2 and Llama-3 (Figure 4a) have more robust alignment training, and the attack does not transfer well, even though the finetuned states would be considered weaker in terms of alignment. This divergence hints at how the adversarial subspace of a model transforms during alignment training. We leave a rigorous analysis of this as a future study.

D ADDITIONAL ITERATION DISTRIBUTIONS

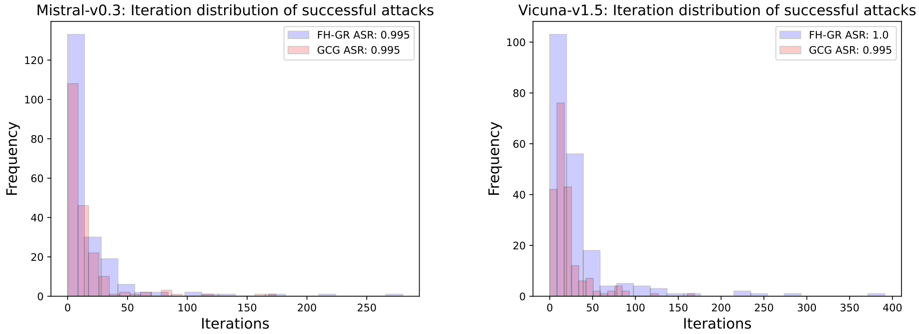
Figure 5 illustrates the iteration distribution for Mistral and Vicuna.



(a) Successes when directly attacking Llama-2 and Llama-3's checkpoints

(b) Successes when directly attacking Mistral and Vicuna's checkpoints

Figure 4: Transferability of successful attacks on the base model to its finetuned parameter states. We find that the attack does not necessarily transfer for all models. This seems to be a function of the "distance" between the states and the alignment training received.



(a) Iteration distribution for Mistral-v0.3

(b) Iteration distribution for Vicuna-v1.5

Figure 5: Iteration distribution for successful attacks. We are able to find adversarial strings far more efficiently than GCG, the closest competing baseline.

E FH LEVEL-SET PLOT

Figure 6 presents a conceptual illustration of the homotopy optimization method, elucidating its underlying principles and operational dynamics. This visual representation provides a more detailed intuition of the method's efficacy in navigating complex optimization landscapes.

F LOSS CONVERGENCE ANALYSIS FOR GCG AND FH-GR

In this section, we consider "hard" samples for Llama-2 and Llama-3 that GCG was unable to jailbreak, but FH-GR (initialized from checkpoint-500) was successful.

Figure 7 shows the change in average loss with iterations throughout the progression of homotopy. We find that easier optimization problems and the solution of the preceeding problem enables a consistently lower loss throughout the homotopy process.

Figure 8 examines how the adversarial strings found on weaker models by Functional Homotopy, affect the average loss on the base model. We see that the loss consistently decreases, indicating that we are able to avoid local optima and successfully jailbreak the model faster.

Figure 9 shows the robustness of Functional Homotopy. Despite initializing the attack with stronger checkpoints (i.e., models fine-tuned for fewer epochs), we still find that loss converges more quickly that GCG and results in a jailbroken response from the model.

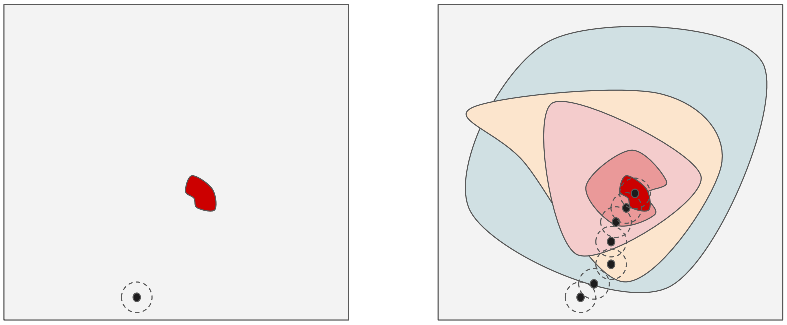
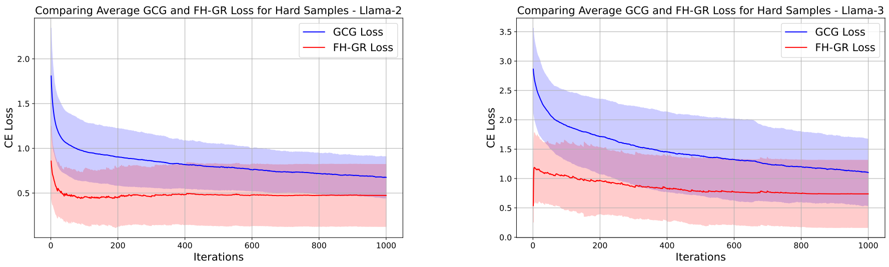
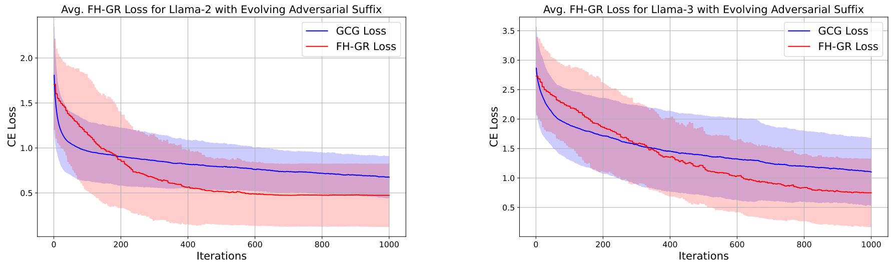


Figure 6: Conceptual illustration of homotopy methods for suffix optimization. (a) Left: Greedy local search heuristic. The red region denotes successful suffixes. The search initiates from a starting point (black solid) and iteratively moves to the optimal neighboring input (dashed circle) based on loss values, potentially leading to local optima entrapment due to non-convexity. (b) Right: Homotopy approach. A series of progressively challenging optimization problems is constructed, with easier problems having larger solution spaces. The solution set gradually converges to that of the original problem. Adjacent problems in this continuum have proximal solutions, facilitating effective neighborhood search. Despite the underlying non-convexity, initiating from a near-optimal point simplifies each problem-solving step.



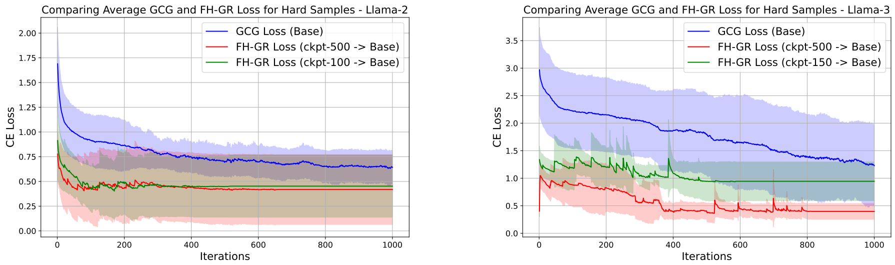
(a) FH-GR loss for Llama-2, across checkpoints (b) FH-GR loss for Llama-3, across checkpoints

Figure 7: A loss comparison of GCG and FH-GR on "hard" samples. We see that homotopy starts off with a substantially smaller loss, due to the misalignment process. We initialize FH-GR from checkpoint 500. As we iterate and successfully jailbreak an intermediate model, we replace it (as described in Algorithm 1), until we reach the base model by iteration 1000. We further note in Section 5 that FH-GR converges more quickly than GCG.



(a) FH-GR loss for base Llama-2 (b) FH-GR loss for base Llama-3

Figure 8: A loss comparison of GCG and FH-GR on "hard" samples. Unlike Figure 7, we look at the usefulness of the adversarial strings found by Functional Homotopy, by applying them on the base model. The loss decreases more consistently, before converging to a lower value overall and successfully jailbreaking the model.



(a) FH-GR loss for Llama-2, across checkpoints (b) FH-GR loss for Llama-3, across checkpoints

Figure 9: A loss comparison of GCG and FH-GR initialized from different checkpoints. We take 25 "hard" samples and initialize FH-GR from earlier checkpoints that are more aligned. Of the cases where starting from the earlier checkpoint succeeds (loss in green), we see that FH-GR is still able to converge to a lower loss than GCG. Note that GCG fails on all these cases, where as FH-GR (ckpt-500→base) succeeds on all cases and FH-GR starting from earlier (stronger) checkpoints succeeds on 13 cases for Llama-2, and 6 cases for Llama-3.