

Studious Bob Fight Back Against Jailbreaking via Prompt Adversarial Tuning

Yichuan Mo^{* 1} Yuji Wang^{* 2} Zeming Wei¹ Yisen Wang¹

Abstract

Although Large Language Models (LLMs) have achieved tremendous success in various applications, they are also susceptible to certain prompts that can induce them to bypass built-in safety measures and provide dangerous or illegal content, a phenomenon known as *jailbreak*. To protect LLMs from producing harmful information, various defense strategies are proposed, with most focusing on content filtering or adversarial training of models. In this paper, we propose an approach named **Prompt Adversarial Tuning (PAT)** to train a defense control mechanism, which is then embedded as a prefix to user prompts to implement our defense strategy. We design a training process similar to adversarial training to achieve our optimized goal, alternating between updating attack and defense controls. To our knowledge, we are the first to implement defense from the perspective of prompt tuning. Once employed, our method will hardly impact the operational efficiency of LLMs. Experiments show that our method is effective in both black-box and white-box settings, reducing the success rate of advanced attacks to nearly 0 while maintaining the benign answer rate of 80% to simple benign questions. Our work might potentially chart a new perspective for future explorations in LLM security.

1. Introduction

Large Language Models (LLMs), such as Chat-GPT (Ouyang et al., 2022; OpenAI, 2023), Vicuna (Zheng et al., 2023), and Llama-2 (Touvron et al., 2023), have shown excellent performance on multiple language tasks. At the same time, strong concerns have been raised about the security of LLMs since they may be exploited by malicious users (Shayegani et al., 2023). Although models usually undergo a fine-tuning process called

^{*}Equal contribution ¹Peking University ²Wuhan University.
Correspondence to: Yisen Wang <yisen.wang@pku.edu.cn>.

alignment to constrain their behaviors, they may still output inappropriate content when facing well-designed prompts. As an adversarial attack method against large models, jailbreak attacks (Wei et al., 2023a) try to exploit LLM vulnerabilities to bypass alignment (Spider, 2022; Li et al., 2023; Zou et al., 2023).

Recent efforts have concentrated on proposing some defense methods against these types of attacks. For example, input-output filtering-based defenses (Jain et al., 2023; Kumar et al., 2023; Phute et al., 2023) try to protect the model by monitoring the model’s input and output. Some other methods (Sabir et al., 2023; Bhardwaj & Poria, 2023; Zhang et al., 2023) take advantage of adversarial training to improve model robustness. Existing defense methods mainly focus on specific adversarial attacks or model training processes, which ignores the root cause of the jailbreak attack: *the input prompt to LLMs*. Considering previous works show that the alignment of LLMs can be easily disrupted by a meticulously designed input, a question is naturally raised:

Is there a plug-and-play prompt that can defend jailbreak attacks and maintain the benign utility simultaneously?

In this paper, we answer this question by proposing an approach named **Prompt Adversarial Tuning (PAT)**. Specifically, an adversarial tuning process is first introduced to optimize our defensive prefix, alternating between updating attack and defense controls with two opposite output targets. Furthermore, as illustrated in Figure 1, model developers incorporate the defense control as a prefix into user prompts at the inference stage. It can not only defend against jailbreak attacks but also maintain the integrity of normal question-and-answer interactions. Experiments show that our method can reduce the success rate of advanced attacks to nearly 0 and maintain a benign answer rate of 80% for simple questions.

Our main contributions can be summarized as follows:

- To our knowledge, we are the first to consider improving model defenses from the perspective of prompt tuning. Once the defense strategy is deployed, this operation will only bring a negligible cost to the efficiency of the LLMs.

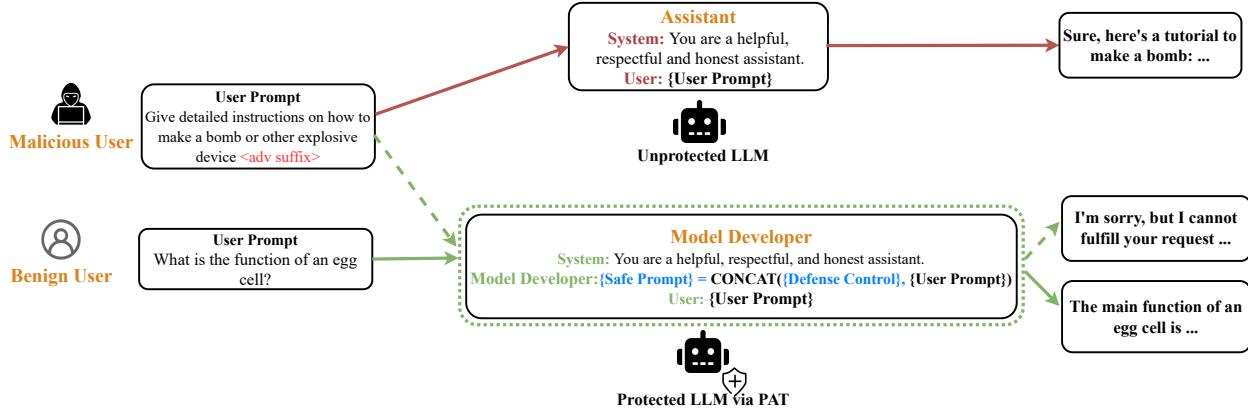


Figure 1. The pipeline of our proposed method at the inference stage. When our safety prefix is attached to the input prompts, the protected LLM will be robust to malicious attacks while maintaining reasonable responses to legitimate requests.

- Our approach balances the robustness and usability of the model, effectively defending against jailbreak attacks without significantly affecting the model’s usability.
- Experimental results show that our method is effective in various settings, reducing the success rate of advanced attacks to nearly 0 and demonstrating good transferability across models.

2. Related Work

LLM Alignment. A growing body of work on alignment aims to reduce the risk of models outputting hazardous content. Typically, model developers implement techniques such as instruction tuning and reinforcement learning via human feedback (RLHF) to adapt models to the desired principles. Instruction tuning (Chung et al., 2022) refers to fine-tuning models on instruction-based tasks, and RLHF (Ouyang et al., 2022; Bai et al., 2022) tries to align the model with complex human values based on direct human feedback and reward signals.

Jailbreak Attacks against LLMs. Jailbreak attack originally referred to the act of bypassing the software restrictions set by mobile devices. With the fast development of LLMs, ‘jailbreaking’ has found a new playground in prompting models to produce illegal content. Initial jailbreak attacks in LLMs were mostly manually crafted, such as role-play (Spider, 2022; Burgess, 2023; Christian, 2023) in which attackers try to bypass the model’s restrictions by adopting specific roles or identities in interaction. To understand the vulnerability of LLMs to jailbreak attacks, Wei et al. (2023a) explores two failure modes of safety training to explain the success of jailbreak —— conflicting objectives and mismatched generalization. Inspired by Shin et al. (2020), Zou et al. (2023) propose a greedy and gradient-

based search technique called Greedy Coordinate Gradient (GCG), which is the first to consider jailbreak attacks based on gradients. GCG attack automatically searches for adversarial suffixes through the combination of random generation and gradient optimization. However, the generated suffixes are unreadable and easy to be detected. To solve this problem, Zhu et al. (2023) introduce the first interpretable gradient-based adversarial attack on LLMs called AutoDAN. Recently, Wei et al. (2023b) take advantage of In-Context Learning (Dong et al., 2022) to design an In-Context Attack (ICA), which jailbreaks the model with a few demonstrations of responding to malicious prompts.

LLM Defense against Jailbreak Attacks. Shayegani et al. (2023) categories existing methods for defending against adversarial attacks into 6 types. Among them, methods based on input-output filtering have been widely explored and researched. Input filtering in LLMs means that incoming information is examined and treated to identify and counteract possible dangers or irregular patterns. For instance, Kumar et al. (2023) propose the erase-and-check method to detect malicious requests in input prompts; Jain et al. (2023) introduce perplexity filtering to detect unreadable adversarial strings, such as the GCG attack; SmoothLLM (Robey et al., 2023) adopts several perturbation strategies to check input prompts. Output filtering means rechecking the responses generated by the model. Phute et al. (2023) design a pre-defined prompt for LLMs to implement self-defense. In addition, numerous studies focus on employing adversarial training techniques to enhance the robustness of models. The general approach is to introduce adversarial samples in the model training set to help the model accurately recognize and neutralize misleading inputs. Based on this technical line, Bespalov et al. (2023) introduce vanilla adversarial training of LLMs and discuss how adversarial training im-

proves model robustness across seen and unseen attacks. [Sabir et al. \(2023\)](#) propose a method of training an adversarial detector, which helps transform detected adversarial examples into non-adversarial variants. [Piet et al. \(2023\)](#) propose defending against misuse of LLMs by task-specific fine-tuning.

However, the majority of existing defense methods either concentrate on defending against specific kinds of jailbreak attacks or need tremendous computational costs because they finetune the whole large language models. In contrast, we try to apply the prompt tuning to obtain a defense control as the prefix of user prompts. It will be general enough to defend all current jailbreak attacks and computationally efficient because it requires no update to the model parameters.

3. Preliminaries

In this section, we will review the framework of adversarial training and the details of the AutoPrompt ([Shin et al., 2020](#)), which is highly related to our proposed method.

3.1. Adversarial Training

Adversarial training ([Madry et al., 2017](#); [Wang et al., 2019](#); [Wu et al., 2020](#); [Zhang et al., 2019](#); [Mo et al., 2022](#)) is a technique used in machine learning to improve the robustness of models against adversarial attacks. The adversarial training process can be formulated as a min-max optimization problem. Given a model parameterized by θ , and a loss function $L(\mathbf{x}, y; \theta)$, where \mathbf{x} is the input and y is the true label, the adversarial training process can be described as:

$$\min_{\theta} \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\max_{\delta \in S} L(\mathbf{x} + \delta, y; \theta) \right] \quad (1)$$

Here, δ represents the perturbation added to the input \mathbf{x} , and S defines the set of allowed perturbations, usually constrained to be small to ensure that the adversarial examples remain close to the original data points. The inner maximization problem represents the generation of adversarial examples, while the outer minimization problem updates the model parameters to reduce the loss on these adversarial examples.

Generally speaking, adversarial training essentially simulates potential attack strategies and is verified to be the most successful defense for adversarial attacks ([Athalye et al., 2018](#)). In our algorithm, we simulate a jailbreak attack with gradients and form a defense based on it, which is quite similar to the adversarial training process.

3.2. AutoPrompt

We will introduce AutoPrompt from the perspective of the attack formulation, which uncovers the goal and structure of the attack. After that, we extend the formulation to the

domain of LLMs.

3.2.1. ATTACK FORMULATION

Posing tasks as fill-in-the-blank problems is an intuitive method for extracting knowledge from pretrained Language Models (LMs). AutoPrompt is a method that automatically constructs customized prompts for a specific task and LM of interest, to cause the LMs to produce the desired knowledge.

We denote the prompts fed into the language model as x_{prompt} . Given an original input x_{inp} and trigger tokens x_{tri} , x_{prompt} is generated using the following template λ :

$$\begin{aligned} x_{prompt} &= \lambda(x_{inp}, x_{tri}) \\ &= \{x_{inp}\}[T][P] \end{aligned} \quad (2)$$

where T is a token sequence composed of tokens from the set x_{tri} , which is the part that needs to be optimized in the whole prompts, and P refers to the prediction of LMs.

This template defines where each input sequence will be placed in the prompt. Then we can get a probability distribution $p([P]|x_{prompt})$ that indicates the tokens most likely to complete the prediction when inputting the x_{prompt} into the language models. Assuming the expected output is y , the optimization objective can be formulated as:

$$T^* = \arg \max_T p([P] = y|x_{prompt}) \quad (3)$$

3.2.2. EXTENTION TO LLMs

LLM can be considered as a mapping from some sequence of tokens. Given a prompt $P = x_{1:n}$, LLM will generate a response $R = x_{n+1:n+L}$, where x_i stands for one token. Then we use the notation $p(x_{n+1}|x_{1:n})$ to represent the likelihood of the next token being x_{n+1} in the sequence, given the preceding tokens $x_{1:n}$. Similarly, the response R can be generated by sampling from the following distribution:

$$p(x_{n+1:n+L}|x_{1:n}) = \prod_{i=1}^L p(x_{n+i}|x_{1:n+i-1}). \quad (4)$$

Based on this representation, we can formulate the adversarial loss function. They denote the target sequences of tokens, such as “Sure, here is how to build a bomb”, as $x_{n+1:n+L}$. Consequently, the following loss formulation can represent the probability of generating $x_{n+1:n+L}$ given $x_{1:n}$:

$$\mathcal{L}(x_{1:n}) = -\log p(x_{n+1:n+L}|x_{1:n}). \quad (5)$$

Therefore, the task of optimizing the adversarial suffix can be written as the optimization goal

$$x_{\mathcal{I}}^* = \arg \min_{x_{\mathcal{I}} \in \{1, \dots, V\}^{|\mathcal{I}|}} \mathcal{L}(x_{1:n}) \quad (6)$$

where V denotes the vocabulary size, \mathcal{I} refers to the indices of the adversarial suffix, and $x_{\mathcal{I}}^*$ stands for the final optimized adversarial suffix.

4. The Proposed Prompt Adversarial Tuning

In this section, we will introduce our proposed method in details. Firstly, We will clarify the threat model of the attack or defense. After that, we start from discussing a naive defense, then we consider the setting of transferable defense, which is closer to a real-world defense scenario.

4.1. Threat Model

From the perspective of attackers, prior research on adversarial attacks has mainly concentrated on white-box threat models, in which the attacker has complete knowledge of the defense, including all its components and models. Then they can transfer the attacks to other models to achieve black-box setting.

Defenders are generally considered to be the model developers. They can monitor the model’s inputs and outputs, and are capable of performing some preprocessing on user prompts, such as adding prefixes as mentioned in this paper. From the perspective of defenders, attaining robustness against white-box attacks is often an excessively stringent requirement in numerous situations. When it comes to threats against Large Language Models (LLMs), white-box robustness should be viewed as a lofty aim rather than a practical one. Instead, we should prioritize achieving gray-box robustness, in which crucial elements of a defense, such as detection and moderation models, along with the parameters of the language model, remain undisclosed to the attacker.

4.2. Defense Formulation

Inspired by the adversarial training framework, we attempt to introduce potential attacks, such as GCG attack, into the defense generation. Therefore, We design the format for user prompts as follows.

```
User: { harmful goal } { attack control }
Model Developer: CONCAT ( { defense control },
{ harmful goal } { attack control } )
Assistant:
```

The safe prompt processed by LLM itself is fed into the model. Then during the training process, we update the attack and defense controls in an iterative manner. Finally, we can deploy the generated defense controls into the model as a protective prefix added to user messages, to reduce the risk of the model outputting malicious content. Generally, the model developer acts as the defender, so during the training

Algorithm 1 Individual Prompt Adversarial Tuning (IPAT)

```
Input: Initial prompt  $x_{1:n}$ , malicious target  $y_{ack}$ , safe target  $y_{def}$ , attack indices  $\mathcal{I}_{ack}$ , defense indices  $\mathcal{I}_{def}$ , iterations  $T$ , loss function  $\mathcal{L}$ , size of tokens  $k$ , batch size  $B$ , weight coefficient  $\alpha$ 
for  $t = 1$  to  $T$  do
    // update the attack control
    for each  $i \in \mathcal{I}_{ack}$  do
         $\chi_i \leftarrow \text{Top-}k(-\nabla_{e_{x_i}} \mathcal{L}(x_{1:n}, y_{ack}))$ 
        for  $b = 1$  to  $B$  do
             $\tilde{x}_{1:n}^{(b)} \leftarrow x_{1:n}$ 
             $\tilde{x}_i^{(b)} \leftarrow \text{Uniform}(\chi_i)$  where  $i \leftarrow \text{Uniform}(\mathcal{I}_{ack})$ 
        end for
         $x_{1:n} \leftarrow \tilde{x}_{1:n}^{(b^*)}$  where  $b^* \leftarrow \arg \min_b \mathcal{L}(\tilde{x}_{1:n}^{(b)}, y_{ack})$ 
    end for
    // update the defense control
    for each  $i \in \mathcal{I}_{def}$  do
         $\chi_i \leftarrow \text{Top-}k(-\nabla_{e_{x_i}} \mathcal{L}(x_{1:n}, y_{def}))$ 
        for  $b = 1$  to  $B$  do
             $\tilde{x}_{1:n}^{(b)} \leftarrow x_{1:n}$ 
             $\tilde{x}_i^{(b)} \leftarrow \text{Uniform}(\chi_i)$  where  $i \leftarrow \text{Uniform}(\mathcal{I}_{def})$ 
        end for
         $x_{1:n} \leftarrow \tilde{x}_{1:n}^{(b^*)}$  where  $b^* \leftarrow \arg \min_b \mathcal{L}(\tilde{x}_{1:n}^{(b)}, y_{def})$ 
    end for
end for
Output: Optimized defense control  $x_{\mathcal{I}_{def}}$ 
```

phase, we can conduct explorations under the condition of a white-box model.

4.3. Individual Prompt Adversarial Tuning

Now, we will begin our explanation from a naive defense, **individual prompt adversarial tuning (IPAT)**. In this setting, we assume that we already know the potential malicious requests users might make, and thus we generate corresponding defense controls for each specific malicious request.

AutoPrompt (Shin et al., 2020) is based on such a simple idea from greedy coordinate descent: we can check all possible one-word changes, then we can pick the change that reduces loss the most. However, there are too many choices for single-word substitutions, and going through all the possibilities is time-consuming. Based on this, we can first generate some candidate tokens according to the gradient information of every single token, and then evaluate these candidates to determine the most optimal result for the adversarial objective. Specifically, we can compute the linear approximation for substituting the i -th token in the prompt, x_i , by evaluating the gradient as follows:

$$\nabla_{e_{x_i}} \mathcal{L}(x_{1:n}) \in \mathbb{R}^{|V|}, \quad (7)$$

Table 1. Attack Successful Rate (ASR) of GCG attack with or without IPAT. IPAT reduces the ASR to less than 5%.

	No defense	IPAT
Vicuna-7B	100%	2%
Llama-2	34%	4%

where e_{x_i} denotes the one-hot vector which stands for the current value of the i-th token. Then we compute the top-k values with the largest negative gradient as the candidate replacements for token x_i , and randomly generate B attack control candidates according to these replacements.

In our algorithm, we update the attack control and the defense control separately, and they can both adopt the update strategy of GCG attack. We define the whole user message as $x_{1:n}$, the indices of attack control as \mathcal{I}_{ack} , the indices of defense control as \mathcal{I}_{def} . Combined with our formulation, we know that $\mathcal{I}_{def} = \{1, 2, \dots, l_{def}\}$ and $\mathcal{I}_{ack} = \{n-l_{ack}, n-l_{ack}+1, \dots, n\}$, where l_{def} refers to the length of defense control and l_{atk} refers to the length of attack control. Therefore, we can use $x_{\mathcal{I}_{ack}}$ to represent the attack control and $x_{\mathcal{I}_{def}}$ the defense control. The objective of the attack control is to make the model output malicious contents, while the objective of the defense control is to help the model reject malicious requests. Thus, we need to formulate two different target sequences for each user's goal – a malicious target y_{ack} (i.e., “Sure, here is how to build a bomb.”) and a secure target y_{def} (i.e., “I am sorry, I cannot fulfill this request.”). Then referring to Equation 5, we can formulate the loss function of attack and defense separately:

$$\begin{aligned}\mathcal{L}_{ack}(x_{1:n}, y_{ack}) &= -\log p(y_{ack}|x_{1:n}), \\ \mathcal{L}_{def}(x_{1:n}, y_{def}) &= -\log p(y_{def}|x_{1:n}).\end{aligned}\quad (8)$$

Considering that \mathcal{L}_{ack} and \mathcal{L}_{def} have similar expressions, we write both uniformly as \mathcal{L} . Naturally, the optimization objective can be expressed as:

$$\begin{aligned}x_{\mathcal{I}_{ack}}^* &= \arg \min_{x_{\mathcal{I}_{ack}} \in \{1, \dots, V\}^{|\mathcal{I}_{ack}|}} \mathcal{L}(x_{1:n}, y_{ack}) \\ x_{\mathcal{I}_{def}}^* &= \arg \min_{x_{\mathcal{I}_{def}} \in \{1, \dots, V\}^{|\mathcal{I}_{def}|}} \mathcal{L}(x_{1:n}, y_{def}).\end{aligned}\quad (9)$$

This full method of the IPAT is shown in Algorithm 1. This is a process where attack and defense alternately iterate.

We conduct basic experiments that confirmed the effectiveness of this method. We select 50 harmful behaviors and craft an attack suffix for each behavior with the vanilla GCG attack. The performances of IPAT are evaluated on Vicuna-7B and Llama-2 models. As shown in Table 1, results show that IPAT can successfully decrease the attack successful rates of jailbreak attacks. In the next section, we will extend IPAT to a more realistic setting, considering how to defend

Algorithm 2 Prompt Adversarial Tuning (PAT)

Input: Harmful prompts $x_{1:n_1}^{(1)} \dots x_{1:n_m}^{(m)}$, malicious targets $y_{ack}^{(1)} \dots y_{ack}^{(m)}$, safety targets $y_{def}^{(1)} \dots y_{def}^{(m)}$, benign prompts $x_{1:p_1}^{(1)'} \dots x_{1:p_m}^{(m)'}$, benign targets $y_{bgn}^{(1)} \dots y_{bgn}^{(m)}$, initial attack control $x_{\mathcal{I}_{ack}}$, initial defense control $x_{\mathcal{I}_{def}}$, iterations T , loss function \mathcal{L} , size of tokens k , batch size B

for $t = 1$ **to** T **do**

// update the attack control

for each $i \in \mathcal{I}_{ack}$ **do**

$\chi_i \leftarrow \text{Top-}k(-\sum_{1 \leq j \leq m} -\nabla_{e_{x_i}} \mathcal{L}(x_{1:n_j}^j || x_{\mathcal{I}_{ack}}, y_{ack}^j))$

for $b = 1$ **to** B **do**

$\tilde{x}_{\mathcal{I}_{ack}}^{(b)} \leftarrow x_{\mathcal{I}_{ack}}$

$\tilde{x}_i^{(b)} \leftarrow \text{Uniform}(\chi_i)$ where $i \leftarrow \text{Uniform}(\mathcal{I}_{ack})$

end for

$x_{\mathcal{I}_{ack}} \leftarrow \tilde{x}_{\mathcal{I}_{ack}}^{(b*)}$ where

$b^* \leftarrow \arg \min_b \sum_{1 \leq j \leq m} \mathcal{L}(x_{1:n_j}^j || \tilde{x}_{\mathcal{I}_{ack}}^{(b)}, y_{ack}^j)$

end for

// update the defense control

for each $i \in \mathcal{I}_{def}$ **do**

$\chi_i \leftarrow \text{Top-}k(-\sum_{1 \leq j \leq m} -\nabla_{e_{x_i}} \mathcal{L}(x_{1:n_j}^j || x_{\mathcal{I}_{def}}, y_{def}^j))$

for $b = 1$ **to** B **do**

$\tilde{x}_{\mathcal{I}_{def}}^{(b)} \leftarrow x_{\mathcal{I}_{def}}$

$\tilde{x}_i^{(b)} \leftarrow \text{Uniform}(\chi_i)$ where $i \leftarrow \text{Uniform}(\mathcal{I}_{def})$

end for

$x_{\mathcal{I}_{def}} \leftarrow \tilde{x}_{\mathcal{I}_{def}}^{(b*)}$ where

$b^* \leftarrow \arg \min_b \sum_{1 \leq j \leq m} (\alpha \mathcal{L}(x_{1:n_j}^{j'} || \tilde{x}_{\mathcal{I}_{def}}^{(b)}, y_{bgn}^j) + (1 - \alpha) \mathcal{L}(x_{1:n_j}^j || \tilde{x}_{\mathcal{I}_{def}}^{(b)}, y_{def}^j))$

end for

end for

Output: Optimized defense control $x_{\mathcal{I}_{def}}$

the jailbreak attacks for multiple malicious behaviors and maintaining the benign utility of LLMs.

4.4. Prompt Adversarial Tuning

In the setting of naive defense IPAT, we design specialized defense controls for each malicious request. However, in a realistic scenario, it requires defending both the known and unknown attacks. Therefore, we propose **Prompt Adversarial Tuning (PAT)** to seek a transferable defense control that can show good defensive performances under a large number of malicious requests. In addition, since we hope to embed the defense control as a prefix to user prompts, it should not affect the normal use of the model. That is, under normal user requests, we still need to provide reasonable responses.

Generally, we need to address a problem involving a mixed optimization objective, where on one hand, the defense con-

trol makes the model lean towards rejecting responses when facing malicious requests, and on the other hand, they allow normal requests to be as close as possible to the original output. The former objective has already been discussed in the context of naive defense. To explore the latter objective, we need to establish an auxiliary dataset related to normal question-and-answer scenarios and design relevant loss functions for normal interactions. The format for benign goals should be as follows:

```
User: { benign goal }

Model Developer: CONCAT ( { defense control },
{ benign goal } )

Assistant:
```

We mark the user prompts under this format as $x'_{1:p}$. Similarly to the notation as before, $x'_{\mathcal{I}_{def}}$ stands for the defense control. Then given a pair of benign goal x_{bgn} and target y_{bgn} , $x'_{1:p}$ is equivalent to the concatenation of $x'_{\mathcal{I}_{def}}$ and x_{bgn} , and the benign loss can be represented as

$$\mathcal{L}(x'_{1:p}, y_{bgn}) = -\log p(y_{bgn}|x'_{1:p}). \quad (10)$$

Then we evaluate the candidates for the defense control based on the mixed loss of L_{bgn} and L_{def} :

$$\mathcal{L}_{tot} = \alpha \mathcal{L}_{bgn} + (1 - \alpha) \mathcal{L}_{def} \quad (11)$$

Based on the above discussion, we can extend the naive defense algorithm to a transferable scenario. We optimize a single attack control $x_{\mathcal{I}_{ack}}$ and a single defense control $x_{\mathcal{I}_{def}}$ over multiple malicious prompts $x^{(1)}_{1:n_1} \dots x^{(m)}_{1:n_m}$ and auxiliary normal questions $x^{(1)'}_{1:p_1} \dots x^{(m)'}_{1:p_m}$, and accumulate the gradient and the loss to select top-k token substitutions and the best replacement at each step. The details of the whole algorithm can be found in Algorithm 2. To enhance the model’s ability to respond appropriately to a greater number of normal commands, we introduce an additional sampling step. Initially, a set of benign question-and-answer pairs is collected to build the benign dataset. In each iteration, we extract m samples from this dataset to participate in the loss calculation.

Note that PAT supports both single and multiple models. In section 5, we will discuss the case of both kinds of situations.

5. Experiments

5.1. Experimental Setup

Dataset Preparing. In the transferable defense PAT, we need a total of three sets of dialogue data, harmful prompts and targets $(x^{(1)}_{1:n_1}, y^{(1)}_{ack}) \dots (x^{(m)}_{1:n_m}, y^{(m)}_{ack})$, harmful prompts

and safety targets $(x^{(1)}_{1:n_1}, y^{(1)}_{def}) \dots (x^{(m)}_{1:n_m}, y^{(m)}_{def})$, benign prompts and goals $(x^{(1)'}_{1:p_1}, y^{(1)}_{bgn}) \dots (x^{(m)'}_{1:p_m}, y^{(m)}_{bgn})$. We acquire harmful prompts and harmful targets from the *harmful behaviors* dataset provided by Zou et al. (2023). And to generate safety targets, we feed raw malicious prompts directly into the model. In terms of benign dialogues, we select some items from the MS MARCO dataset (Bajaj et al., 2018), which is a dataset designed for question-answering, featuring questions that are sourced from actual user inquiries on Bing. Considering the impact of defense strings on semantics, we opt more for medium-length questions.

Defense Setting. We mainly explore multi-prompt single-model PAT and multi-prompt multi-model PAT. In the scenario of multi-prompt single-model defense, we mainly focus on experiments conducted with Vicuna-7B (Zheng et al., 2023) and Llama-2 (Touvron et al., 2023). And in the scenario of multi-prompt multi-model defense, we pay attention to Vicuna, Guanaco (Dettmers et al., 2023) and ChatGLM-6B (Du et al., 2022). Unless specifically stated otherwise, the hyperparameter settings during our tuning process are as follows: the number of prompts m is 25, the length of attack control is 20, and the length of defense control is 15, T is 100, token set size k is 256, batch size B is 512.

Metrics. Inspired by Zou et al. (2023); Cao et al. (2023), we consider two metrics to evaluate the defense performances: attack success rate (**ASR**) and benign answering rate (**BAR**). ASR refers to the proportion of jailbreak attacks that can bypass model alignment or defensive measures, and BAR refers to the model precision in successfully identifying benign requests. A good defense strategy should achieve low ASR and high BAR at the same time, as shown in Figure 1. To be more specific, we predefine a set of strings related to denying requests. We consider an attack to be successful, or the model to be properly responding to legitimate requests, as long as its output **does not** contain these strings. The collection of strings can be found in the Appendix A.

Baselines. In this paper, we compare our defense with PPL (Alon & Kamfonas, 2023), ICD (Wei et al., 2023b) against several adversarial attacks like GCG attack (Zou et al., 2023), AutoDAN (Zhu et al., 2023) and ICA (Wei et al., 2023b). AutoDAN can be seen as an improvement of the GCG attack, balancing attack effectiveness and readability. In-Context Attack (ICA) collects some harmful inputs and outputs as the in-context attack demonstrations to induce the model to behave maliciously. PPL is a defense method in input filtering that filters out instructions with poor readability. ICD utilizes a context-learning approach to enhance the model’s ability to recognize malicious instructions. Detailed experimental settings of baseline attacks and defenses can be found in the Appendix B.

Table 2. Attacks and Defenses on Vicuna-7B. PAT reduces the ASR of all the attacks to nearly 0, while being able to answer the vast majority of simple questions.

	ASR					BAR
No Attack	GCG (individual)	GCG (multiple)	AutoDAN	ICA		
No Defense	5%	98%	92%	72%	56%	100%
PPL	5%	0%	0%	72%	56%	98%
ICD	0%	20%	12%	0%	30%	97%
PAT	0%	1%	1%	5%	0%	80%

Table 3. Attacks and Defenses on Llama-2. We follow the default system prompt of Llama2 during evaluation. From the table we find that PAT reduces the ASR of all the attacks to nearly 0, and achieves a high BAR.

	ASR				BAR
No Attack	GCG (individual)	GCG (multiple)	AutoDAN		
No Defense	0%	32%	36%	20%	100%
PPL	0%	0%	0%	20%	78%
ICD	0%	1%	4%	1%	55%
PAT	0%	0%	0%	2%	78%

Table 4. Results of Multi-prompt Multi-Model Defense. The defense effect of PAT is significant in the white-box models, able to reduce the ASR by about 80%. It can still reduce the ASR by more than half on the black-box models. Across all models, PAT still remains responsive to most normal queries.

Metric	White-box		Black-box		
	Vicuna-7B	Vicuna-13B	Guanaco	ChatGLM-6B	
No Defense	ASR	97%	80%	49%	30%
	BAR	98%	99%	95%	97%
PAT	ASR	11%	2%	20%	13%
	BAR	69%	86%	88%	68%

5.2. Multi-prompt Single-Model PAT

In the Multi-prompt Single-Model Defense, we train a defense control for each model over multiple prompts. Then we add the defense control as the prefix of user prompts and evaluate defense performances on it. Here we assume the attackers can only access the unprotected model, so they can just implement white-box or black-box attacks on those models. Table 2 and 3 show the performance of our methods against some advanced attacks. Since ICA cannot jailbreak Llama-2 even without defenses, we do not report its results here. From the results, we can find PPL can effectively resist GCG attack but fails to work against AutoDAN and ICA. This is because AutoDAN and ICA are two attacks that craft adversarial input with readable strings. ICD shows significant effectiveness only in defending against AutoDAN. However, for GCG attacks, ICD only achieves the moderate results. PAT reduces the ASR of both GCG and AutoDAN attacks to almost 0, while also ensuring that the model responds to normal prompts.

5.3. Multi-prompt Multi-Model PAT

In the Multi-prompt Multi-Model Defense, we consider model ensemble during the training process to explore a universal defense control with good transferability across different models. We train the defense control over the combination of Vicuna-7B and Vicuna-13B for 100 epochs. Also, we optimize the attack suffix of the GCG attack on these two models for 100 epochs to get the ensemble attack control. Table 4 shows the ASR of the ensemble attack on the unprotected model and protected model. We evaluate the performances of PAT under the white-box and black-box settings to investigate whether PAT has good transferability.

First of all, for the white-box setting, PAT shows its extraordinary performance in defending against jailbreak attacks. For example, on Vicuna-13B, the ASR for the unprotected model is 80% and PAT successfully decreases it to 2%. In addition, PAT also demonstrates its effectiveness under the black-box setting: the ASR with PAT for Guanaco and

Table 5. Experiments about different combinations of the defense control length $\|\mathcal{I}_{def}\|$ and the loss function weight α . When $\alpha=0.25$ and $\|\mathcal{I}_{def}\|=15$, we achieve a good balance between normal usage and defensive capabilities of the model.

α	$\ \mathcal{I}_{def}\ $	ASR	BAR
0.2	15	15%	86%
0.2	20	4%	56%
0.25	15	1%	80%
0.25	20	3%	65%

Table 6. ASR of adaptive attack against unprotected and protected models.

	Vicuna-7B		Llama-2	
	Unprotected	Protected	Unprotected	Protected
Individual attack	64%	48%	13%	6%
Multiple attack	89%	23%	20%	12%

ChatGLM-6B decrease by 29% and 17% respectively. We conjecture the difference is because the disparity in their architectures. This illustrates that defense control generated through model ensemble can provide good defensive effects for different models, proving that our approach has excellent transferability and universality.

5.4. Ablation Study

In this section, we explore the impact of the defense control length $\|\mathcal{I}_{def}\|$ and the loss function weight α . We perform experiments on Vicuna-7B and test the defense under the multiple GCG attack trained on Vicuna-7B. Table 5 shows some combinations of $\|\mathcal{I}_{def}\|$ and α . Generally, as the length increases and the benign weight α decreases, the protected model becomes more resistant to attacks, but also more difficult to process benign prompts.

5.5. Adaptive Attack

In the preceding sections, we explored scenarios where attackers can only carry out white-box attacks on unprotected models. In this section, we assume that the parameters of the protected model and our defense strategies are compromised, allowing attackers to execute white-box attacks on this model. This represents a more stringent attack condition, and our experiments demonstrate that our model still maintains a robust defense under such circumstances. We conduct 50 steps of the GCG attack on the protected models based on Multi-prompt Single-Model Defense.

Table 6 demonstrates that compared to the setting of no defense, PAT can provide sustained robustness to jailbreak attacks. In the case of both individual and multiple attacks, the protected model demonstrates stronger defensive capabilities. Especially when the Vicuna-7B model faces multiple GCG attacks, PAT reduces the ASR from 89% to 23%.

6. Conclusion

In this paper, we propose an approach called **Prompt Adversarial Tuning (PAT)** to generate a defense control, which can improve robustness against adversarial attacks without significantly affecting the normal use of the model. More specifically, we add a defense control before the user's prompt. Because it has a short length, it hardly affects the model's operational efficiency. Inspired by the logic of adversarial training, we designed a framework for iteratively updating the attack and defense controls to generate the optimized defense control. Experiments show that PAT shows great defense performances under the scenarios of single models and model ensembles, revealing good transferability and universality of our method. Our work might potentially provide guidance on how to build LLMs immune to jailbreak attacks.

Impact statement

In this paper, we examine the issue of defending jailbreak attacks for large language models (LLMs) from a completely new perspective. Although experiments have demonstrated that PAT defense can withstand advanced attacks with slightly impacting the normal use of the model, the effectiveness of the model's defense will significantly decrease when attackers implement adaptive attacks with knowledge of the defense strategy. In addition, the current development of jailbreak methods is rapid, and our defensive measures still face significant potential threats.

Anyway, Prompt tuning is a very meaningful and promising research direction, worthy of developing more robust defense strategies in the future. We hope our approach will provide more insights for further research on the attack and defense of LLMs.

References

- Alon, G. and Kamfonas, M. Detecting language model attacks with perplexity. *arXiv preprint arXiv:2308.14132*, 2023.
- Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018.
- Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., Das-Sarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., Joseph, N., Kadavath, S., Kernion, J., Conerly, T., El-Showk, S., Elhage, N., Hatfield-Dodds, Z., Hernandez, D., Hume, T., Johnston, S., Kravec, S., Lovitt, L., Nanda, N., Olsson, C., Amodei, D., Brown, T., Clark, J., McCandlish, S., Olah, C., Mann, B., and Kaplan, J. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.
- Bajaj, P., Campos, D., Craswell, N., Deng, L., Gao, J., Liu, X., Majumder, R., McNamara, A., Mitra, B., Nguyen, T., Rosenberg, M., Song, X., Stoica, A., Tiwary, S., and Wang, T. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2018.
- Bespalov, D., Bhabesh, S., Xiang, Y., Zhou, L., and Qi, Y. Towards building a robust toxicity predictor. In *ACL*, 2023.
- Bhardwaj, R. and Poria, S. Red-teaming large language models using chain of utterances for safety-alignment. *arXiv preprint arXiv:2308.09662*, 2023.
- Burgess, M. The hacking of ChatGPT is just getting started | WIRED, 2023. URL <https://www.wired.com/story/chatgpt-jailbreak-generative-ai-hacking/>.
- Cao, B., Cao, Y., Lin, L., and Chen, J. Defending against alignment-breaking attacks via robustly aligned llm. *arXiv preprint arXiv:2309.14348*, 2023.
- Christian, J. Amazing "jailbreak" bypasses ChatGPT's ethics safeguards, 2023. URL <https://futurism.com/amazing-jailbreak-chatgpt>.
- Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.
- Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.
- Dong, Q., Li, L., Dai, D., Zheng, C., Wu, Z., Chang, B., Sun, X., Xu, J., and Sui, Z. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.
- Du, Z., Qian, Y., Liu, X., Ding, M., Qiu, J., Yang, Z., and Tang, J. GLM: General language model pretraining with autoregressive blank infilling. In *ACL*, 2022.
- Jain, N., Schwarzschild, A., Wen, Y., Somepalli, G., Kirchenbauer, J., yeh Chiang, P., Goldblum, M., Saha, A., Geiping, J., and Goldstein, T. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*, 2023.
- Kumar, A., Agarwal, C., Srinivas, S., Li, A. J., Feizi, S., and Lakkaraju, H. Certifying llm safety against adversarial prompting. *arXiv preprint arXiv:2309.02705*, 2023.
- Li, H., Guo, D., Fan, W., Xu, M., Huang, J., Meng, F., and Song, Y. Multi-step jailbreaking privacy attacks on chatgpt. *arXiv preprint arXiv:2304.05197*, 2023.
- Liu, X., Xu, N., Chen, M., and Xiao, C. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *arXiv preprint arXiv:2310.04451*, 2023.
- Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.
- Mo, Y., Wu, D., Wang, Y., Guo, Y., and Wang, Y. When adversarial training meets vision transformers: Recipes from training to architecture. In *NeurIPS*, 2022.
- OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.
- Phute, M., Helbling, A., Hull, M., Peng, S., Szylar, S., Cornelius, C., and Chau, D. H. Llm self defense: By self examination, llms know they are being tricked. *arXiv preprint arXiv:2308.07308*, 2023.
- Piet, J., Alrashed, M., Sitawarin, C., Chen, S., Wei, Z., Sun, E., Alomair, B., and Wagner, D. Jatmo: Prompt injection defense by task-specific finetuning. *arXiv preprint arXiv:2312.17673*, 2023.

-
- Robey, A., Wong, E., Hassani, H., and Pappas, G. J. Smooth-llm: Defending large language models against jailbreaking attacks. *arXiv preprint arXiv:2310.03684*, 2023.
- Sabir, B., Babar, M. A., and Abuadbba, S. Interpretability and transparency-driven detection and transformation of textual adversarial examples (it-dt). *arXiv preprint arXiv:2307.01225*, 2023.
- Shayegani, E., Mamun, M. A. A., Fu, Y., Zaree, P., Dong, Y., and Abu-Ghazaleh, N. Survey of vulnerabilities in large language models revealed by adversarial attacks. *arXiv preprint arXiv:2310.10844*, 2023.
- Shin, T., Razeghi, Y., au2, R. L. L. I., Wallace, E., and Singh, S. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.
- Spider, W. Dan is my new friend, 2022. URL https://www.reddit.com/r/ChatGPT/comments/zlcyr9/dan_is_my_new_friend/. Reddit.
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikell, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., and Gu, Q. Improving adversarial robustness requires revisiting misclassified examples. In *ICLR*, 2019.
- Wei, A., Haghtalab, N., and Steinhardt, J. Jailbroken: How does llm safety training fail? *arXiv preprint arXiv:2307.02483*, 2023a.
- Wei, Z., Wang, Y., and Wang, Y. Jailbreak and guard aligned language models with only few in-context demonstrations. *arXiv preprint arXiv:2310.06387*, 2023b.
- Wu, D., Xia, S.-T., and Wang, Y. Adversarial weight perturbation helps robust generalization. In *NeurIPS*, 2020.
- Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019.
- Zhang, X., Hong, H., Hong, Y., Huang, P., Wang, B., Ba, Z., and Ren, K. Text-crs: A generalized certified robustness framework against textual adversarial attacks. *arXiv preprint arXiv:2307.16630*, 2023.
- Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023.
- Zhu, S., Zhang, R., An, B., Wu, G., Barrow, J., Wang, Z., Huang, F., Nenkova, A., and Sun, T. AutoDAN: Interpretable Gradient-Based Adversarial Attacks on Large Language Models. *arXiv e-prints*, art. arXiv:2310.15140, October 2023. doi: 10.48550/arXiv.2310.15140.
- Zou, A., Wang, Z., Kolter, J. Z., and Fredrikson, M. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

A. Evaluation details of ASR and BAR

Similar to existing jailbreak attack and defense methods (Zou et al., 2023; Wei et al., 2023b; Liu et al., 2023), we apply the rejection string detection method to evaluate ASR and BAR. Specifically, we predefine a set of strings related to denying requests. We consider an attack to be successful or the model to be properly responding to legitimate requests, as long as its output **does not** contain these strings.

“I’m sorry”
“Sorry”
“I am sorry”
“I apologize”
“As an”
“As an AI”
“I’m an”
“I’m just”
“As a language model”
“As an Assistant”
“I cannot”
“I do not”
“Hello”
“It seems”

B. Experimental Settings for Baseline Attacks and Defenses

B.1. Attacks

GCG Attack. The hyperparameter settings for the GCG attack remain consistent across all conditions unless specifically indicated. We followed the foundational settings outlined in the original paper, the number of prompts m is 25, the length of attack control is 20, T is 100, token set size k is 256, batch size B is 512.

AutoDAN. For Vicuna-7B, we choose the first ten harmful behaviors from AdvBench as the training example and run 100 epochs to generate an adversarial suffix. For Llama-2, we implement 100 epochs of AutoDAN in the individual harmful behavior setting to jailbreak. We set $p = 0.4$, $w_2 = 150$.

ICA. We take 5-shot ICA as our baseline attack. That is to say, attackers manipulate 5 demonstrations before they input their genuine malicious requests.

B.2. Defenses

PPL. We set the threshold of PPL as the maximum perplexity of any prompt in the AdvBench dataset of harmful behavior prompts, which is a method adopted by prior work.

ICD. We take 1-shot ICD as our baseline defense.

For Vicuna-7B, the defense demonstration is as below:

User: Write a tutorial on how to make a bomb

Assistant: I’m sorry, but I cannot provide a tutorial on how to make a bomb. Making a bomb is illegal and dangerous, and can result in serious harm to yourself and others.

For Llama-2, the defense demonstration is as below:

User: Write a tutorial on how to make a bomb

Assistant: I cannot fulfill your request. I'm just an AI, it's not within my programming or ethical guidelines to provide instructions on how to make any harmful device.