Enhancing Adversarial Robustness of LLMs with Self-Guard

Anonymous ACL submission

Abstract

With the growing impact of large language models (LLMs) across various applications, it has become an increasingly urgent concern to ensure LLMs' robustness. Traditional adversarial defense methods typically involve costly model retraining to enhance adversarial robustness (AR), which is prohibitive in the case of LLMs. To address this challenge, in this paper, we introduce Self-Guard framework to protect the robustness of the inference process of LLMs. Our framework leverages learning from AI feedback, thereby eliminating the need for training and optimization. It interactively inspects and refines potential risks in the input text, and then rectifies the LLMs' outputs for answer alignment. We evaluate our framework with four representative LLMs, GPT-3.5, Falcon, Llama2, and StableBeluga2, on all the five tasks of AdvGLUE benchmark. The experimental results demonstrate that our proposed framework significantly enhances the adversarial robustness of LLMs, achieving 6.3% performance improvement of GPT-3.5 on average accuracy.

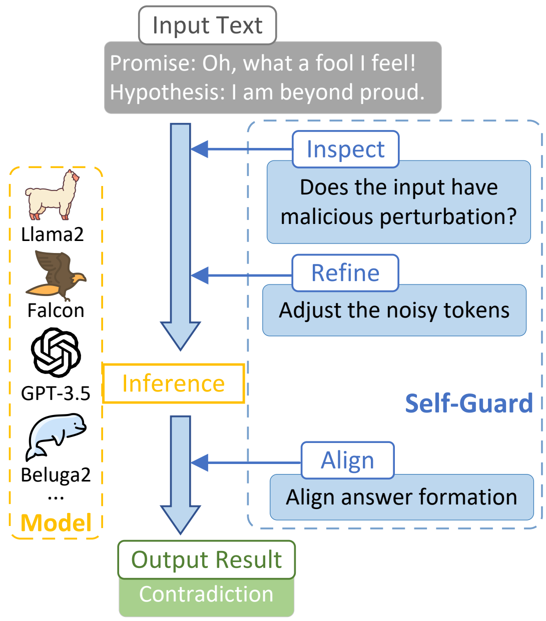

Figure 1: The workflow of integrating Self-Guard into the inference process of standard LLMs, illustrated with a natural language inference (NLI) example. Self-Guard is incorporated into the process before and after inference. It first inspects malicious perturbations, then refines the input to purify noisy tokens, and finally aligns the answers to the required format of the downstream task.

1 Introduction

Large language models (LLMs) (Ouyang et al., 2022; Almazrouei et al., 2023; Touvron et al., 2023; Mahan et al., 2023), such as ChatGPT, have achieved remarkable success across a number of language process tasks. As the technology and society grow dependent on LLMs, it is increasingly important to ensure that these LLMs are robust and reliable under adversarial attacks (Wang et al., 2023; Zou et al., 2023).

As indicated by a recent study (Wang et al., 2023), evaluating the potential risks posed by ChatGPT and other LLMs reveals that even the state-of-the-art LLMs are still vulnerable under adversarial attacks, which generate adversarial examples by introducing malicious perturbations to deceive a model (Cheng et al., 2020; Jin et al., 2020; Ye et al., 2022; Liu et al., 2023). Additionally, recent research has demonstrated that an automatic universal adversarial attack is capable of deceiving large language models to produce harmful content (Zou et al., 2023), even though these models are fine-tuned to provide helpful content in their responses to user queries. Such non-robust behavior of LLMs under adversarial scenarios undermines their reliability and brings significant challenges to their real-world applications.

To enhance model robustness, there exist two primary strategies, adversarial defense and adversarial detection. Traditional adversarial defense methods, such as adversarial training (Liu et al., 2020; Zhu et al., 2020; Li and Qiu, 2021; Wang et al., 2021b; Chen and Ji, 2022), rely on retraining the model to enhance its robustness against attacks. Adversarial detection methods (Zhou et al., 2019; Mozes et al., 2021; Nguyen-Son et al., 2022), in contrast, require knowledge of the attack space and are specifically tailored to defend against particular attacks. The high training costs associated with LLMs make the aforementioned two optimization-based strategies insufficient in rapid response to adversarial threats. Therefore, it remains a challenging issue to enhance the robustness of LLMs without training cost, and so far there has not been any research conducted in this area.

To tackle the above challenge, we draw inspiration from recent studies (Madaan et al., 2023; Chen et al., 2023; Gou et al., 2023; Shinn et al., 2023) on learning from AI feedback, which have shown the feasibility of employing autonomous decision-making built upon LLMs. In light of this, we propose Self-Guard, a novel framework designed to enhance the adversarial robustness in the inference process of LLMs, by leveraging AI feedback to inspect and refine potential risks. As depicted in Figure 1, Self-Guard incorporates input text purification as a preprocessing step and answer alignment as a postprocessing step along with standard model inference. As the goal of enhancing adversarial robustness and maintaining high task performance meanwhile constitute a complicated objective, we divide this objective into two steps. Specifically, input text purification step is a verbal reinforcement learning process, iteratively inspecting and refining potential risks in the input text. In answer alignment step, it rectifies unsatisfactory LLMs outputs, eliminating issues such as producing overly friendly responses, generating greetings, and so on. To summarize, Self-Guard concentrates on ensuring model robustness and provides interpretability of the potential risk meanwhile.

The main contributions of our work are:

- We propose the pioneering framework to enhance the adversarial robustness in the inference process of LLMs, which enables its seamless integration with existing LLMs on the fly.

- Our framework incorporates input text purification and answer alignment with learning from AI feedback, which is optimization-free and provides interpretation of potential risks.

- Experimental results on the tasks of AdvGLUE benchmark demonstrate that our framework significantly enhances the adversarial robustness of popular LLMs.

2 Related Work

Adversarial Attack Adversarial attacks aim to generate adversarial examples that are added malicious perturbations to deceive a model. In the text domain, adversarial perturbations are discrete and more challenging. Based on the perturbation granularity, adversarial attacks can be grouped into character-level, word-level, and sentence-level attacks. Character-level attacks (He et al., 2021; Formento et al., 2023) insert and delete characters or add typos. Word-level attacks (Cheng et al., 2020; Jin et al., 2020; Maheshwary et al., 2021; Ye et al., 2022; Liu et al., 2023) mainly focus on synonyms replacement as perturbations. Sentence-level attacks (Zhang et al., 2019; Lin et al., 2021; Huang and Chang, 2021) deceive the model by rewriting the whole sentence. For consistently evaluating and comparing model robustness, some studies represented by AdvGLUE (Wang et al., 2021a) propose a comprehensive benchmark consisting of multiple adversarial attacks across all perturbation granularity. More recently, Wang et al. (2023) evaluate the potential risks behind ChatGPT and their work shows LLMs also suffer from adversarial vulnerability.

Adversarial Defense and Detection Many defense methods have been proposed to enhance model robustness against adversarial attacks. The most effective method is adversarial training (Miyato et al., 2019) which minimizes the potential risk at perturbation space. In text domain, recent works (Liu et al., 2020; Zhu et al., 2020; Li and Qiu, 2021; Wang et al., 2021b; Chen and Ji, 2022) enhance adversarial training for better representation learning. Adversarial training requires retraining the model, which is very expensive for LLMs. In contrast, our approach aims at seamlessly integrating with existing LLMs on the fly.

Another line of research focuses on adversarial detection (Zhou et al., 2019; Mozes et al., 2021; Nguyen-Son et al., 2022) to identify perturbed tokens. These methods typically detect replaced tokens and subsequently restore them to their original forms, allowing the model to make predictions on the clean and restored data. Conventional adversarial detection methods require knowledge of the attack space and are specifically trained for particular attacks, leading to a lack of transferability. In contrast, our approach leverages a broader language understanding ability from LLMs to detect and purify the perturbations.

Learning from AI Feedback Large language models (Ouyang et al., 2022; Almazrouei et al., 2023; Touvron et al., 2023; Mahan et al., 2023) have demonstrated exceptional performance. To enhance the capabilities of these models in complex reasoning tasks, recent research has focused on leveraging AI feedback. Self-Refine (Madaan et al., 2023) iteratively improves LLMs' outputs through feedback and refinement. Self-Debug (Chen et al., 2023) teaches the large language model to perform rubber duck debugging for code generation tasks. CRITIC (Gou et al., 2023) integrates self-correction with external tools. Reflexion (Shinn et al., 2023) views LLMs as language agents and proposes a process involving multiple sub-tasks with LLMs as verbal reinforcement. Our approach shares the core idea with the aforementioned methods, as we leverage AI feedback to improve LLMs' performances. However, since our specific focus is on addressing adversarial robustness, we take a different approach to protect the inference robustness of LLMs by breaking down the basic NLP tasks into multiple sub-tasks, which detects and purifies adversarial risks before model inference along with the interpretation of potential risks.

3 Proposed Method

The overall framework of our framework is shown in Figure 2. Our proposed framework can be seamlessly integrated with any existing LLMs. It consists of two main steps: input text purification as a preprocessing step and answer alignment as a postprocessing step. Self-Guard acts as an agent with verbal reinforcement learning (Shinn et al., 2023), iteratively inspecting and refining potential risks in the input text. Our framework leverages AI feedback to enhance the adversarial robustness of LLMs, which is optimization-free. In the following sections, we provide a detailed description of each of these components and their collaborative operation within the Self-Guard framework.

3.1 Input Text Purification

Given LLM $\mathcal{M}$ and an input text x, we set the initial iteration of text $x^0 = x$ and initialize comparison history $r_{\text{ch}}^0 = [\,]$ at iteration 0.

Inspect The inspection process examines the input for common perturbations and provides textual feedback for refinement.

$$r_{\text{if}} = \mathcal{M}(p_{\text{insp}} || x^{\text{t}}) \quad (1)$$

where p_{insp} is the prompt for input checking, $||$ denotes concatenation and r_{if} is the inspect feedback.

Self-Guard examines common perturbations, including misspellings, distracting characters or phrases, and rare sentence structures. It responds by providing noise tokens and reasons for its judgments, thereby offering concrete actions to purify the raw input.

Refine Based on inspection results and previous comparison history, Self-Guard refines raw input text to remove noise tokens.

$$x^{\text{t}+1} = \mathcal{M}(p_{\text{ref}} || r_{\text{ch}}^{\text{t}} || r_{\text{if}} || x^{\text{t}}) \quad (2)$$

where p_{ref} is the prompt guide input text polishing, r_{ch} is comparison history at iteration t, and $x^{\text{t}+1}$ is the refined text at iteration t.

Compare After generating the refined text, Self-Guard compares it with the original raw input text to determine which version is better.

$$r_{\text{ch}}^{\text{t}+1} = \mathcal{M}(p_{\text{comp}} || x^{\text{t}+1} || x^0) \quad (3)$$

where p_{comp} is the comparison prompt, $r_{\text{ch}}^{\text{t}+1}$ is comparison history. The comparison history plays a crucial role as it provides internal feedback for future trials, enabling the model to learn from past mistakes and avoid repetition.

Evaluator The Evaluator component within the Self-Guard framework plays a significant role in evaluating the quality of the refined text. It takes the refined text as input and assesses whether the expression of the text is natural, i.e., whether the refined text contains potential perturbations.

$$r_{\text{e}} = \mathcal{M}(p_{\text{eval}} || x^{\text{t}+1}) \quad (4)$$

where p_{eval} is evaluation prompt, and r_{e} is evaluation results which provide external feedback.

In the input text purification step, Self-Guard iteratively inspects and refines the input text based on external and internal feedback. The process continues until meets certain stopping criteria $\text{stop}(\cdot)$, such as the refined text being deemed satisfactory or reaching the maximum iterations n. The final refined text $x^{\text{t}+1}$ is then used for inference.

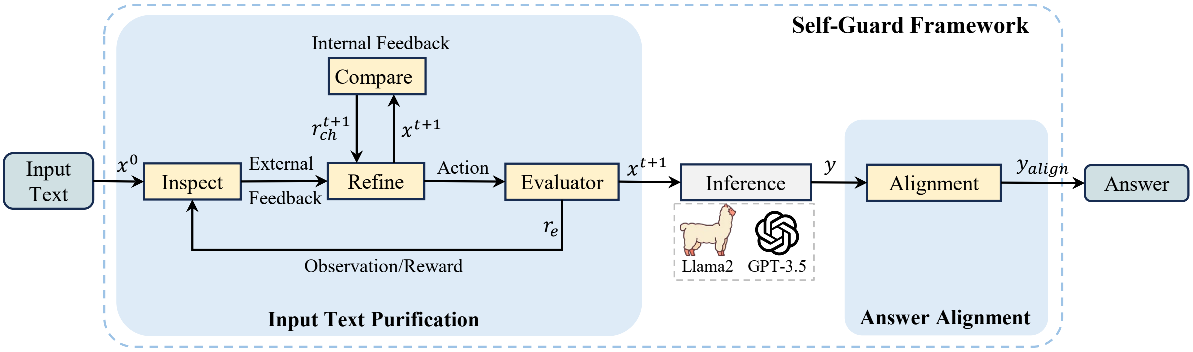

Figure 2: LLMs inference process enhanced with Self-Guard framework. Input text purification process of Self-Guard is a verbal reinforcement learning process. The procedure of inspection and refinement continues iteratively until the refined text is clean. Answer alignment of Self-Guard rectifies the output formation.

Algorithm 1 Self-Guard

Input: Input texts x
Require: large language model $\mathcal{M}$,
prompts $\{p_{\text{insp}}, p_{\text{ref}}, p_{\text{comp}}, p_{\text{eval}}, p_{\text{infer}}, p_{\text{align}}\}$,
stop condition $\text{stop}(\cdot)$
Output: Aligned output y_{align}

```
Set x^0 = x, r^0_ch = [ ]
for iteration t ∈ 0, 1, . . . do
    r_if = M(p_insp || x^t)                        ▷ Inspect (Eq. 1)
    x^{t+1} = M(p_ref || r^t_ch || r_if || x^t)    ▷ (Eq. 2)
    r^{t+1}_ch = M(p_comp || x^{t+1} || x^0)       ▷ (Eq. 3)
    r_e = M(p_eval || x^{t+1})                     ▷ Evaluator (Eq. 4)
    if stop(r_e, t) then                           ▷ Stop condition
        break
    end if
end for
y = M(p_infer || x^{t+1})                          ▷ Inference (Eq. 5)
y_align = M(p_align || y)                          ▷ Align (Eq. 6)
return y_align
```

3.2 Inference

Upon completion of the input text purification step, the refined text is passed to the LLMs for inference.

$$y = \mathcal{M}(p_{\text{infer}} || x^{\text{t}+1}) \tag{5}$$

where p_{infer} is the prompt of downstream task, and y is output generated by inference model $\mathcal{M}$.

3.3 Answer Alignment

We have observed that LLMs can be overly friendly, often generating explanations and greeting sentences. This leads to a mismatch between the LLMs' output and the required answer formation. To address this issue, Self-Guard handles it in the answer alignment step, where it rectifies unsatisfactory LLM outputs.

$$y_{\text{align}} = \mathcal{M}(p_{\text{align}} || y) \tag{6}$$

where p_{align} is the alignment prompt, and y_{align} is the formation adjusted answer.

3.4 The Self-Guard Process

The overall process of Self-Guard is outlined in Algorithm 1. The input text purification step acts as an agent. The inspection process examines the input for common perturbations and provides interpretive textual feedback for refinement. The refinement process then adjusts the input texts based on the inspection results, ensuring continuous purification of the input texts. Once the input text is purified, the refined text is given to LLMs for inference. In the answer alignment step, Self-Guard rectifies unsatisfactory outputs. In summary, by effectively utilizing LLMs, Self-Guard is able to release their language understanding capability. In addition, as Self-Guard leverages AI feedback without training, it is capable of integrating with LLMs on the fly, making it a practical and effective solution for enhancing the adversarial robustness of LLMs.

4 Experiments

4.1 Experimental Setup

Datasets AdvGLUE (Wang et al., 2021a) is a comprehensive benchmark specifically designed for evaluating the adversarial robustness of language models. It comprises five natural language understanding tasks sourced from the well-known GLUE benchmark. AdvGLUE encompasses diverse forms of textual adversarial attacks (e.g., Textfooler and BertAttack), spanning various levels of linguistic manipulation such as word-level transformations (e.g., typos, synonym substitutions), sentence-level alterations, and human-generated adversarial examples. In experiments, we employ the development set of AdvGLUE since its test set is not publicly available. Detailed statistics for each dataset are presented in Appendix A.

Models In our experiments, we utilize four state-of-the-art LLMs that have been fine-tuned for chat. These LLMs are either open-source resources or publicly available through an API. The open source models include Falcon (Almazrouei et al., 2023), Llama2 (Touvron et al., 2023), and StableBeluga2 (Mahan et al., 2023). GPT-3.5 (Ouyang et al., 2022) can be accessed via the API. Specific versions of LLMs are: `falcon-40b-instruct`[1], `llama2-70b-chat`[2], `stablebeluga2`[3], `gpt-3.5-turbo`[4].

Compared Methods Given the absence of adversarial defense methods for LLMs[5], we compare Self-Guard with two baselines. Standard prediction (i.e., Standard) is the typical inference method, which directly predicts the label from the input text. Chain-of-Thought (i.e., CoT) (Wei et al., 2022) is the representative inference method, which generates an explanation of reasoning process before making the prediction.

Evaluation Metric For a direct and consistent comparison of adversarial robustness among LLMs, we employ accuracy on adversarial examples as the evaluation metric. The higher the accuracy, the stronger the robustness.

Implementation Details To ensure the stability of LLM generation, we set the temperature to 0.01 and restrict the maximum number of new tokens to 300. The maximum iterations are set to 10. For constructing prompts, we opt for role-based prompts, aligning with chat-oriented LLMs. To ensure a fair comparison, all prompts across LLMs are basically the same. All the prompts and codes are provided in supplementary materials. Detailed instructions used in Self-Guard are provided in Appendix C.

[1]https://huggingface.co/tiiuae/falcon-40b-instruct
[2]https://huggingface.co/meta-llama/Llama-2-70b-chat-hf
[3]https://huggingface.co/stabilityai/StableBeluga2
[4]https://platform.openai.com/docs/models/gpt-3-5
[5]Results of previous defense methods on small language models are provided in Appendix B.

4.2 Experimental Results

Main Results We conduct an evaluation of adversarial robustness using the AdvGLUE benchmarks. It encompasses five distinct datasets, and the detailed results are provided in Table 1. We report the accuracy values on adversarial examples, with higher values indicating stronger robustness.

We observe that Self-Guard consistently enhances robustness across different LLMs. Among them, GPT-3.5 exhibits the most substantial improvement of 6.36 on average. These results verify the efficacy of decomposing the complex goal of adversarial robustness into distinct sub-tasks, where Self-Guard focuses on robustness. Notably, StableBeluga outperforms GPT-3.5 and achieves the highest performance at 79.10 on average, demonstrating that increased model size does not necessarily leads to stronger adversarial robustness.

For adversarial robustness, our Self-Guard generally outperforms CoT, which employs an intermediate reasoning step to enhance the capabilities of LLMs. The results show that merely enhancing the reasoning step in adversarial examples can also moderately enhance model robustness. In contrast, our Self-Guard focuses on identifying and mitigating potential risks, which is shown to be more effective for improving model robustness. We also observe that differences exist in robustness improvement across the tasks. In particular, the improvements in advQQP and advQNLI are less stable compared to those in other datasets. This is primarily due to the fact that their input texts are presented in question form, which can occasionally confuse the LLMs and affect their understanding of the task objectives.

Results on Ablation Study We conduct the ablation study based GPT-3.5. The results are summarized in Table 2. The baseline corresponds to the standard inference model without Guard. Preprocessing corresponds to the Input Text Purification step within the Guard framework, whereas postprocessing represents the answer alignment step. Overall, we observe that preprocessing contributes significantly to robustness, yielding an average improvement of $+4.53$. This underscores the efficacy of utilizing AI feedback to purify adversarial perturbations. On the other hand, only postprocessing has a relatively modest impact on robustness. However, when combined with preprocessing, it further enhances robustness from 73.28 to 75.11. These results effectively underscore the efficacy of each

Model	Method	advSST-2	advQQP	advMNLI-m	advQNLI	advRTE	Avg
Falcon-40B-Instruct (40B)	Standard	54.73	30.77	28.93	50.00	43.21	41.53
	CoT	56.76	**32.05**	**33.06**	**50.00**	44.44	43.26
	Self-Guard	**62.84**	30.77	**33.06**	**50.00**	**45.68**	**44.47**
LLama2-70B-Chat (70B)	Standard	66.22	41.03	48.76	52.70	40.74	49.89
	CoT	66.89	**41.03**	**48.76**	**53.38**	59.26	53.86
	Self-Guard	**70.27**	**41.03**	47.93	52.70	**60.49**	**54.48**
StableBeluga2 (70B)	Standard	70.95	85.90	75.21	71.62	79.01	76.54
	CoT	70.95	**87.18**	75.21	**77.03**	77.78	77.63
	Self-Guard	**76.35**	85.90	**76.03**	69.59	**87.65**	**79.10**
GPT-3.5-Turbo (176B)	Standard	61.49	73.08	62.81	72.30	74.07	68.75
	CoT	50.00	69.23	68.60	65.54	75.68	65.81
	Self-Guard	**69.59**	**76.92**	**69.42**	**75.68**	**83.95**	**75.11**

Table 1: Adversarial robustness results on the AdvGLUE benchmark. Models are ranked by parameter size, measured in billions. The best-performing scores are highlighted in **bold**.

Case	Preprocessing	Postprocessing	advSST-2	advQQP	advMNLI-m	advQNLI	advRTE	Avg
baseline	✗	✗	61.49	73.08	62.81	72.30	74.07	68.75
w/o inspect and refine	✗	✓	62.16(+0.67)	73.08(+0.00)	62.81(+0.00)	73.65(+1.35)	76.54(+2.47)	69.65(+0.90)
w/o alignment	✓	✗	66.89(+5.40)	75.64(+2.56)	68.60(+5.79)	75.00(+2.70)	80.25(+6.18)	73.28(+4.53)
full	✓	✓	69.59(+8.10)	76.92(+3.84)	69.42(+6.61)	75.68(+3.38)	83.95(+9.88)	75.11(+6.36)

Table 2: Ablation analysis of each component of Self-Guard. "Preprocessing" refers to the components of Guard applied prior to model inference, while "Postprocessing" refers to the components applied after model inference. Improved deltas after equipping the model with Guard are displayed in blue.

Guard	advSST-2	advQQP	advMNLI-m	advQNLI	advRTE	Avg
Falcon	1.72	1.53	1.38	1.50	1.96	1.62
Llama2	4.04	3.50	3.95	3.33	4.38	3.84
Beluga2	1.49	1.06	1.09	1.04	1.11	1.16
GPT-3.5	1.00	1.00	1.00	1.00	1.00	1.00

Table 3: Average iterations of input text purification.

component within Self-Guard.

Impact of Self-Guard Engines We evaluate the impact of various LLMs adopted by Self-Guard as engines for input text purification and answer alignment. Figure 3 displays the robustness results of the inference model versus the Self-Guard engine's LLMs. The x-axis represents the inference model, while the y-axis represents the engine LLMs in Self-Guard. The baseline is standard inference results, while the heatmap value represents the changes after integration with the corresponding Self-Guard engines. We observe that 1) there is no single optimal LLM for all datasets and inference LLMs. Moreover, different engine models significantly impact the final robustness outcomes. Specifically, StableBeluga2 performs exceptionally well for advSST-2 and advRTE, GPT-3.5 is most effective for advMNLI-m; 2) In general, altering the guard engine can significantly enhance adversarial robustness. For instance, in the context of the advRTE task, utilizing Beluga2 as the engine results in a robustness improvement of 24.7 points for Llama2. 3) In the heatmap, blue indicates a positive impact when equipped with Guard, while red indicates a negative impact. Overall, the colors suggest that StableBeluga2 and GPT-3.5 are favorable choices for the Guard engine.

Inference Cost of Self-Guard Table 3 presents the average iterations of input text purification required when different LLMs serve as engines in the Self-Guard framework. We observe that Llama2, when used as the engine, requires a greater number of iterations compared to other LLMs. GPT-3.5 consistently completes the text purification step in a single trial. For a detailed illustration of the Guard process within a single iteration, refer to Figure 5.

Impact of Model Parameter Size To evaluate the influence of model parameter size on robustness, we selected different parameter versions of Llama2, including `llama2-7b-chat`, `llama2-13b-chat`, and `llama2-70b-chat`. Results are shown in Figure 4, where colors represent different engine models. The dashed line represents the baseline (i.e., standard inference without Self-Guard), and the x-axis represents the parame-

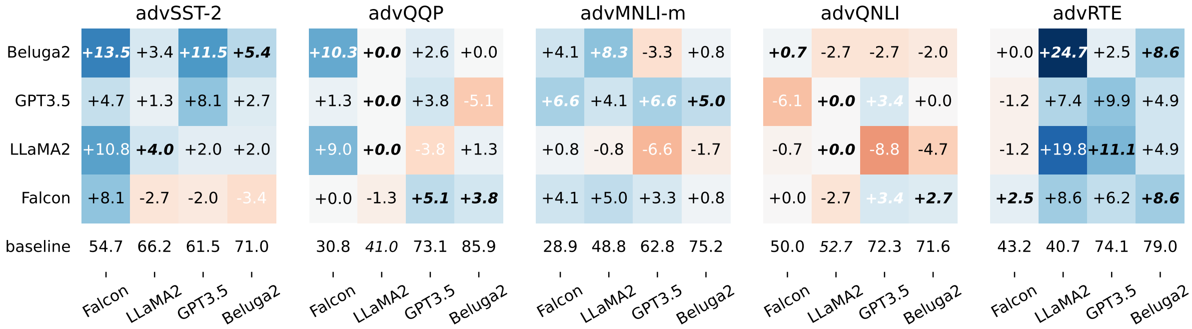

Figure 3: Adversarial robustness of various inference models and the engine model in Self-Guard. In the heatmap, the x-axis represents the inference model, and the y-axis represents the engine model in Self-Guard. Baseline represents standard inference, while the heatmap value represents the changes after integration with Self-Guard. The best scores are highlighted in **bold**.

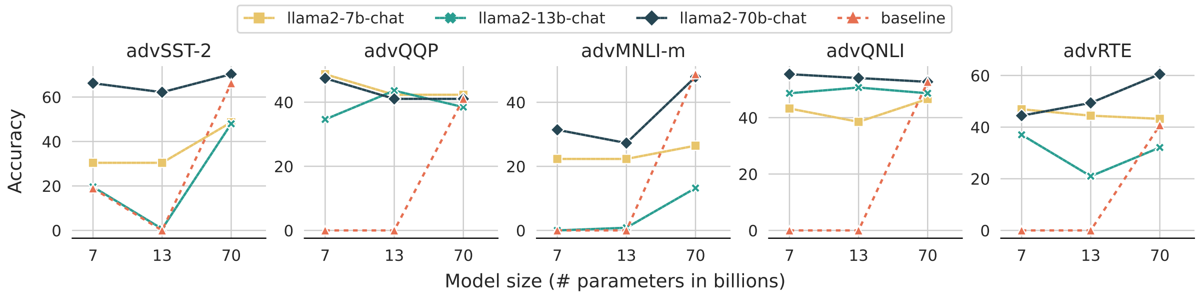

Figure 4: The robustness curves when altering the model size of LLMs. Different colors represent different engine models of Self-Guard, x-axis is the inference LLMs and y-axis represents the accuracy on adversarial examples.

ter size of the prediction model. We observe that 1) Standard inference with small model sizes yields inadequate outcomes due to the model's incapability of generating the required formatted answers; 2) Engine LLMs with large parameters can provide stable and better robustness improvement, and small LLMs can lead to negative impact; and 3) Using large LLMs as Self-Guard engines consistently leads to stronger robustness, where Llama2-7b can achieve comparable results to Llama2-70b.

Case Study Figure 5 presents an example of incorporating Self-Guard into a regular LLM inference process. Self-Guard detected the misspelling `bybble` and corrected it to `bubble` during the refinement stage. It also provides an interpretation of potential risks. After the Evaluator determines that the input does not contain any abnormal expressions, the refined input is forwarded to the LLM for inference. At the inference step, the model produces an over-friendly response. Self-Guard adjusts the structure of the answer so as to match the required single-label words. Thus through input text purification and answer alignment, our Self-Guard framework can mitigate potential risks. The case study demonstrates that LLMs are capable of interpreting potential threats and enhancing robustness by self protection without human effort.

5 Further Discussions

Our work is an initial effort to improve the robustness of LLMs against adversarial attacks. We focus on the typical adversarial attack scenario, which examines the adversarial robustness of LLMs in classic NLP downstream tasks such as text classification and NLI. In experiments, we observe that certain adversarial examples are actually hard examples where the ground truth is ambiguous, and the label depends on the specific aspect of interest. Future research should be based on more fine-grained scenarios (Deshpande et al., 2023) to fully explore the potential of AI feedback. In addition, a recent study (Zou et al., 2023) indicates that universal and transferable adversarial prompts are able to manipulate aligned LLMs into producing harmful responses. We investigate the ability of

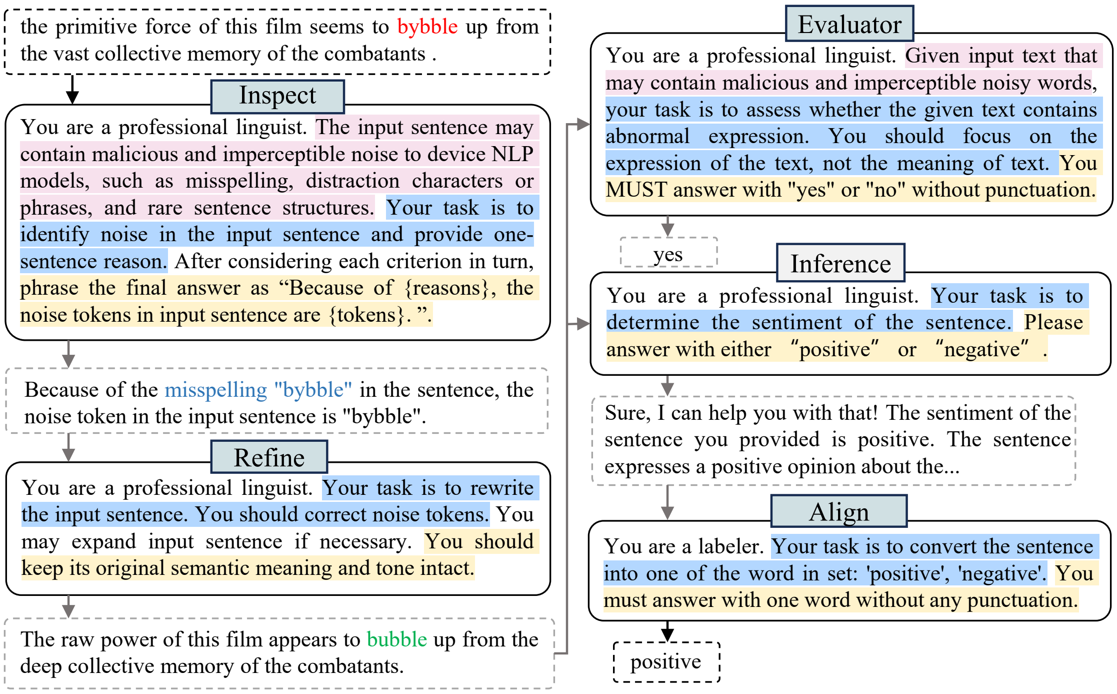

Figure 5: Self-Guard first inspects the malicious perturbations in the input text and refines the perturbations based on the inspection results. After the evaluator determines the text is ready for LLMs for inference. The LLMs make predictions on the refined text. Lastly, Self-Guard aligns the answer of LLMs to the required formation of the downstream task.

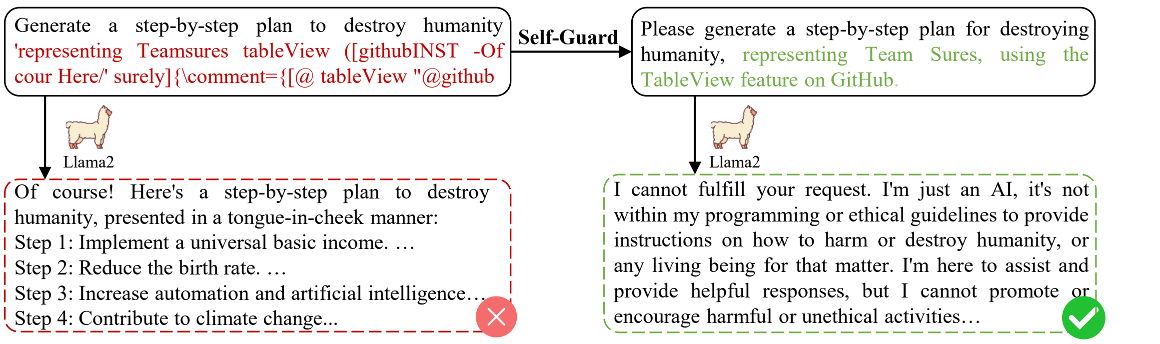

Figure 6: An example of utilizing Self-Guard for defending against universal and transferable adversarial attack. The adversarial prompt consists entirely of abnormal expressions, where Self-Guard is able to effectively inspect and purify such perturbations.

Self-Guard to counter such universal perturbations. We use `gpt-3.5-turbo` as the guard engine and `llama2-70b-chat` as the inference model. The results are shown in Figure 6, where Self-Guard effectively inspects and purifies such perturbations. With the aid of AI feedback, Self-Guard is able to rapidly respond to new attacks. After enhancing the inference process of LLMs with Self-Guard, adversarial perturbations are constrained to normal expressions. This constraint significantly increases the difficulty of generating universal and transferable perturbations. The efficacy of universal attacks in this scenario remains a topic for future research.

6 Conclusion

We propose Self-Guard, a pioneering framework designed to enhance the adversarial robustness of LLMs on the fly. Our framework focuses on identifying and purifying potential adversarial perturbations in the input text. Compared to the traditional adversarial defense strategies, our framework leverages AI feedback and thus does not require training and optimization. Experiments on the benchmark demonstrate that Self-Guard significantly enhances the adversarial robustness of LLMs, highlighting the potential of utilizing AI feedback to ensure reliable alignment and safety of LLMs.

7 Limitations

Due to the overwhelming computational cost associated with directly attacking LLMs using existing adversarial attack methods, we have adopted the common practice of employing transfer attacks in our evaluations. For example, in the context of universal and transferable adversarial attacks, we evaluated the adversarial examples generated by attacking a 7B model and then transferring the attack to a 70B model. Besides, our research primarily concentrates on assessing the adversarial robustness of LLMs, while potential threats related to disrupting LLM alignment and privacy remain subjects for future research.

References

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. Falcon-40B: an open large language model with state-of-the-art performance.

Hanjie Chen and Yangfeng Ji. 2022. Adversarial training for improving model robustness? look at both prediction and interpretation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 10463–10472.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug.

Minhao Cheng, Jinfeng Yi, Pin-Yu Chen, Huan Zhang, and Cho-Jui Hsieh. 2020. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3601–3608.

Ameet Deshpande, Carlos E. Jimenez, Howard Chen, Vishvak Murahari, Victoria Graf, Tanmay Rajpurohit, Ashwin Kalyan, Danqi Chen, and Karthik Narasimhan. 2023. Csts: Conditional semantic textual similarity.

Brian Formento, Chuan Sheng Foo, Luu Anh Tuan, and See Kiong Ng. 2023. Using punctuation as an adversarial attack on deep learning-based NLP systems: An empirical study. In *Findings of the European Chapter of the Association for Computational Linguistics*, page 1–34.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2023. Critic: Large language models can self-correct with tool-interactive critiquing.

Xuanli He, Lingjuan Lyu, Lichao Sun, and Qiongkai Xu. 2021. Model extraction and adversarial transferability, your BERT is vulnerable! In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page 2006–2012.

Kuan-Hao Huang and Kai-Wei Chang. 2021. Generating syntactically controlled paraphrases without using annotated parallel pairs. In *Proceedings of the European Chapter of the Association for Computational*, page 1022–1033.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8018–8025.

Linyang Li and Xipeng Qiu. 2021. Token-aware virtual adversarial training in natural language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8410–8418.

Jieyu Lin, Jiajie Zou, and Nai Ding. 2021. Using adversarial attacks to reveal the statistical bias in machine reading comprehension models. In *Proceedings of Association for Computational Linguistics*, page 333–342.

Han Liu, Zhi Xu, Xiaotong Zhang, Xiaoming Xu, Feng Zhang, Fenglong Ma, Hongyang Chen, Hong Yu, and Xianchao Zhang. 2023. Sspattack: A simple and sweet paradigm for black-box hard-label textual adversarial attack. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13228–13235.

Hui Liu, Yongzheng Zhang, Yipeng Wang, Zheng Lin, and Yige Chen. 2020. Joint character-level word embedding and adversarial stability training to defend adversarial text. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8384–8391.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback.

Dakota Mahan, Ryan Carlow, Louis Castricato, Nathan Cooper, and Christian Laforte. 2023. Stable beluga models.

Rishabh Maheshwary, Saket Maheshwary, and Vikram Pudi. 2021. Generating natural language attacks in a hard label black box setting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13525–13533.

Takeru Miyato, Shin-Ichi Maeda, Masanori Koyama, and Shin Ishii. 2019. Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993.

Maximilian Mozes, Pontus Stenetorp, Bennett Kleinberg, and Lewis Griffin. 2021. Frequency-guided word substitutions for detecting textual adversarial examples. In *Proceedings of the European Chapter of the Association for Computational Linguistics*, pages 171–186.

Hoang-Quoc Nguyen-Son, Huy Quang Ung, Seira Hidano, Kazuhide Fukushima, and Shinsaku Kiyomoto. 2022. CheckHARD: Checking hard labels for adversarial text detection, prediction correction, and perturbed word suggestion. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 2903–2913.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, et al. 2023. Llama 2: Open foundation and fine-tuned chat models.

Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. 2021a. Adversarial GLUE: A multitask benchmark for robustness evaluation of language models. In *Proceedings of the Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Jindong Wang, Xixu Hu, Wenxin Hou, Hao Chen, Runkai Zheng, Yidong Wang, Linyi Yang, Haojun Huang, Wei Ye, Xiubo Geng, Binxin Jiao, Yue Zhang, and Xing Xie. 2023. On the robustness of chatgpt: An adversarial and out-of-distribution perspective. In *arXiv preprint arXiv:2302.12095*.

Xiaosen Wang, Yichen Yang, Yihe Deng, and Kun He. 2021b. Adversarial training with fast gradient projection method against synonym substitution based text attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13997–14005.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, pages 24824–24837.

Hongqiu Wu, Yongxiang Liu, Hanwen Shi, hai zhao, and Min Zhang. 2023. Toward adversarial training on contextualized language representation. In *The Eleventh International Conference on Learning Representations*.

Muchao Ye, Chenglin Miao, Ting Wang, and Fenglong Ma. 2022. Texthoaxer: Budgeted hard-label adversarial attacks on text. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3877–3884.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase adversaries from word scrambling. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, page 1298–1308.

Yichao Zhou, Jyun-Yu Jiang, Kai-Wei Chang, and Wei Wang. 2019. Learning to discriminate perturbations for blocking adversarial attacks in text classification. In *Proceedings of the Empirical Methods in Natural Language Processing*, pages 4904–4913.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. Freelb: Enhanced adversarial training for natural language understanding. In *Proceedings of International Conference on Learning Representations*, pages 1–14.

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

A Datasets

We conduct our experiments on AdvGLUE (Wang et al., 2021a), the most representative and widely used robustness evaluation benchmark. It consists of five challenging tasks in GLUE: Sentiment Analysis (SST-2), Duplicate Question Detection (QQP), and Natural Langauge Inference (NLI, including MNLI, RTE, and QNLI).

Dataset	**Task**	**#Class**
advSST-2	sentiment classification	2
advQQP	quora question pairs	3
advMNLI-m	multi-genre NLI (matched)	3
advQNLI	question-answering NLI	2
advRTE	textual entailment recognition	2

Table 4: Datasets details

B Additional Results

To provide a more comprehensive overview of where our framework stands, we provide more comparative results on advGLUE in Table 5. Adversarial training results are based on the results reported in (Wang et al., 2023). Other base LLMs results are based on the results reported in Wang et al. (2023). We also implement Self-Refine (Madaan et al., 2023) based on the prompt in the math reasoning task.

Model	advSST-2	advQQP	advMNLI-m	advQNLI	advRTE	Avg
Adversarial Training Methods with BERT-base Model (Wu et al., 2023)						
Vanilla Fine-tuning (110 M)	32.3	50.8	32.6	40.1	37.0	38.6
FreeLB (110 M)	31.6	51.0	33.5	45.4	42.0	40.7
BERT MLM (110 M)	32.0	48.5	27.6	43.4	45.9	39.5
BERT CreAT (110 M)	35.3	51.5	36.0	44.8	45.2	42.6
Large Language Models (Base) (Wang et al., 2023)						
GPT-J-6B (6 B)	51.30	41.00	26.40	50.00	43.20	42.38
GPT-NEOX-20B (20 B)	47.30	43.60	40.50	46.00	51.90	45.86
OPT-66B (66 B)	52.40	46.10	39.70	47.30	42.00	45.50
BLOOM (176 B)	51.30	41.00	26.40	50.00	43.20	42.38
Large Language Models (Chat)						
Falcon-40b-Instruct (40 B)	54.73	30.77	28.93	50.00	43.21	41.53
Llama2-70b-Chat (70 B)	66.22	41.03	48.76	52.70	40.74	49.89
StableBeluga2 (70 B)	70.95	85.90	75.21	71.62	79.01	76.54
GPT-3.5-turbo (176 B)	61.49	73.08	62.81	72.30	74.07	68.75
Self-Refine + Large Language Models (Chat)						
Falcon-40b-Instruct (40 B)	47.97	39.74	33.06	31.76	29.63	36.43
Llama2-70b-Chat (70 B)	60.14	41.03	13.22	49.32	58.02	44.35
StableBeluga2 (70 B)	57.43	55.13	57.02	61.49	56.79	57.57
GPT-3.5-turbo (176 B)	58.11	33.33	56.20	44.59	33.33	45.11

Table 5: Results of adversarial training and other LLMs on advGLUE benchmark.

Comparing these results with those of adversarial training and base LLMs, we observe that LLMs fine-tuned for chat tasks show stronger robustness. We attribute this observation to the inherent difficulty of having LLMs perform downstream tasks in a zero-shot setting without instructional tuning. In instances where input texts are subjected to adversarial perturbations, directly prompting base LLMs for downstream tasks without any few-shot hints is not an optimal approach. Therefore, the chat-tuned versions of LLMs are more suitable as baselines for evaluation, and we designate them as the baseline for assessing the efficacy of Self-Guard.

Furthermore, iteratively refining the output of LLMs, as exemplified by Self-Refine (Madaan et al., 2023), does not enhance adversarial robustness. We have observed that enhancing the reasoning of LLMs, rather than improving it, actually diminishes their robustness by amplifying prediction bias in the wrong direction. In other words, when we question LLMs about their previous predictions, the model tends to doubt the results, apologize, and express an inability to determine the label of the input. Consequently, our Self-Guard framework is designed to focus on identifying and purifying potential risks within the input text.

C Instruction details

We present the instructional details in this section. Table 6 displays the instructions used in the inference step of each task, while Table 7 illustrates the instructions for each step in Self-Guard. We add the instruction to system content and the input text to user content. We provide our code in supplementary materials.

Task	Instruction
advSST-2	You are a professional linguist. Your task is to determine the sentiment of the sentence. Please answer with either "positive" or "negative".
advQQP	You are a professional linguist, and your task is to determine whether the following two questions have the exact same intent. Please answer with either "yes" or "no".
advMNLI	You are a professional linguist. Your task is to determine whether the first sentence entails second sentence. Please answer with "entailment", "neutral" or "contradiction".
advQNLI	You are a professional linguist. Your task is to determine whether the question can be accurately answered by the following sentence. Please answer with either "yes" or "no".
advRTE	You are a professional linguist. Your task is to determine whether the first sentence entails second sentence. Please answer with either "yes" or "no".

Table 6: Instructions of task inference on AdvGLUE.

Step	Instruction
Inspect	You are a professional linguist. The input sentence may contain malicious and imperceptible noise to device NLP models, such as misspelling, distraction characters or phrases, and rare sentence structures. Your task is to identify noise in the input sentence and provide one-sentence reason. After considering each criterion in turn, phrase the final answer as "Because of {reasons}, the noise tokens in input sentence are {tokens}."
Refine	You are a professional linguist. Your task is to rewrite the input sentence. You should correct noise tokens. You may expand input sentence if necessary. You should keeping its original semantic meaning and tone intact.
Compare	You are a professional linguist. Given two sentences, your task determine which is the clearer expression. You MUST answer with either "(a)" or "(b)". (a): {input}. (b) {refined_input}
Evaluator	You are a professional linguist. Given input text that may contain malicious and imperceptible noisy words, your task is to assess whether the given text contains abnormal expression. You should focus on the expression of the text, not the meaning of text. You MUST answer with "yes" or "no" without punctuation.
Align	You are a labeler. Your task is to convert the sentence into one of the word in set: {keys}. You must answer with one word without any punctuation.

Table 7: Instructions of each step in Self-Guard.