RobustAlign: A Deeper and Broader Layer Safety Alignment for LLMs

Anonymous ACL submission

Abstract

The deployment of large language models (LLMs) in real-world applications is hindered by persistent vulnerabilities in safety alignment, where existing methods remain susceptible to jailbreak attacks and alignment collapse after fine-tuning. We observed that this vulnerability has two key sources: 1) shallow alignment: alignment training primarily adjusts shallow top-layer parameters while neglecting deeper layers, and 2) the scarcity of safety key neurons and their high overlap with general key neurons. To address these challenges, we propose RobustAlign, which enhances alignment depth and breadth to achieve robust safety alignment through two synergistic innovations: (1) Chain-of-Thought (CoT)-augmented training data, which increases the information entropy of training samples, and (2) Synergistic Gradient Scaling to promote deeper and broader adjustments. Extensive experiments on five LLMs against six jailbreak attacks demonstrate RobustAlign's superiority: it reduces attack success rates (ASR) by 21%–63% compared to state-of-the-art baselines against jailbreak attacks and subsequent fine-tuning, while preserving downstream task accuracy and introducing minimal computational overhead (<3%).

1 Introduction

The rapid advancement of large language models (LLMs) has revolutionized AI applications. However, persistent security vulnerabilities still hinder their deployment in real-world scenarios (Jiang et al., 2024,Chao et al., 2023). Despite extensive efforts in safety alignment, such as reinforcement learning from human feedback (RLHF) (Ouyang et al., 2022) and safe decoding (Xu et al., 2024), existing alignment methods remain vulnerable to jailbreak attacks (Jiang et al., 2024) and suffer from alignment capability collapse after finetuning for subsequent tasks, as shown in Figure 1. These phenomena demonstrate model vulnerabilities and call

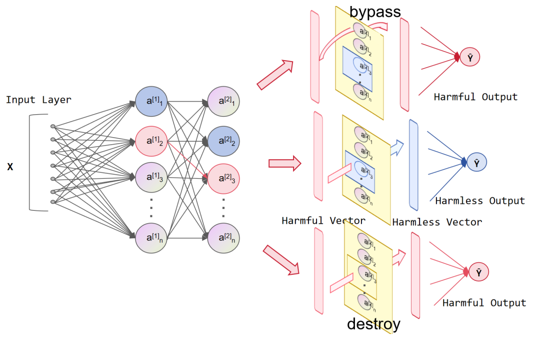


Figure 1: Top: Jialbreak bypasses the safety alignment. Mid: Alignment defense success. Bottom: Subsequent finetuning causes alignment collapse.

for enhanced robustness in safety alignment against jailbreak attacks and fine-tuning-induced failures.

Recent research has focused on the vulnerability of LLM alignment(Qi et al., 2024a). It is posited that this vulnerability may arise from the sparsity of neurons encoding safety knowledge(Wei et al., 2024). Our experiments further corroborate that this sparsity is essentially attributable to the shallow and narrow nature of safety alignment training.

During safety alignment, there is a propensity to adjust the shallow top-layer parameters near the model's output end, while the intermediate layer parameters, which are crucial for semantic understanding and safety cognition, are often overlooked. Due to insufficient adjustment, these intermediate layers are still prone to generating harmful vectors. In contrast, the top layers are given priority for adjustment, leading to the shallowness of alignment training. Meanwhile, within the same layer, there is a tendency to adjust a narrow and concentrated subset of neurons, leading to the narrowness of alignment training. This inter-layer shallowness and intra-layer narrowness collectively contribute to the overall sparsity of safety neurons, which causes the vulnerability of safety alignment. Additionally, our experiment identified a high degree of overlap between safety task neurons and general task neurons. This overlap intensifies the detrimental impact of fine-tuning on safety alignment.

To address these challenges, we propose RobustAlign, which promotes deeper and broader safety alignment training to address the shallowness and scarcity separately. RobustAlign enhances the robustness of alignment capabilities through two modules: 1)**Safety-oriented chain of thought (CoT) augmented dataset**(Wei et al., 2022) increases the information entropy of training samples and the complexity of alignment tasks, which promotes the cross-layer union optimization of multiple neurons and makes alignment training go deeper and broader. 2) **Synergistic Gradient Scaling** adjusts the gradient update amplitude at the layer and neuron level, to address the unavoidable shallowness and narrowness caused by reward hacking. The two mechanisms are complementary, synergistically enhancing the robustness of safety alignment.

Our contributions are three-fold:

Key of vulnerability: Our experiments reveal that the vulnerability of safety alignment stems from the focus on shallow layers during training, as well as the narrow distribution of safety task neurons and their high overlap with general task neurons.

Methodological Progress: Our experimental and theoretical analyses demonstrate that promoting deeper and broader alignment training can enhance the robustness of safety alignment. Meanwhile, utlizing CoT-augmented datasets can facilitate deeper and broader alignment training.

Empirical Verification: Experiments on five LLMs against six jailbreak and three subsequent fintuning attacks show RobustAlign's significant advantages. Compared to baselines, it reduces attack success rates (ASR) by 21% - 63% with minimal computational overhead (<3%) against jailbreak attacks and subsequent fine-tuning.

2 Observational Experiments

In this chapter, we conduct experimental observations and in-depth analysis of the vulnerability of the alignment capability of LLMs. These observations are mainly reflected in two aspects: 1) The vulnerability of alignment arises from the shallowness of alignment training, coupled with the sparsity of safety neurons and their significant overlap with general-purpose neurons; 2) By enhancing the depth and breadth of alignment training, the robustness of alignment capabilities can be significantly improved against jailbreak attacks and fine-tuning. Notably, the use of chain - of - thought (CoT) as training data offers an effective approach to increasing both the depth and breadth of alignment training. In Section 2.1, we investigate the internal causes of the vulnerability of alignment. In Section 2.2, we describe the research on alleviating the vulnerability of alignment.

All the experiments in this section use the Vicuna-13B model as the representative base model and employ the PKU-RLHF as the dataset. We utilize Direct Preference Optimization (DPO) as the alignment method. All experiments are carried out on NVIDIA A100 GPUs with a batch size of 128 samples. Similar properties are also present in other models, which we will introduce in detail in the appendix.

2.1 Cause of Vulnerability

The shallowness of alignment To analyze the vulnerability of alignment, we decode the hidden vectors of all layers into token distribution sequences. This allows us to observe and assess alignment failures during inference when the model is subjected to jailbreak attacks and fine-tuning. The results in Figure 2 show that both scenarios exhibit persistent encoding of harmful preferences in the intermediate layers (layers 14 - 19), while part of the harmful vectors undergo unstable shifts towards harmless in the top layers (layers 29 - 31), but still some vectors remain harmful. This indicates that alignment failure under jailbreak attacks occurs earlier than the alignment taking effect, with harmful variables emerging in the middle layers and alignment capability shift vector to harmless in the top layers. When alignment ability in the top layers fails to shift vectors to harmless, harmful responses become inevitable. Furthermore, the experiment shows excessive alignment phenomena in the top layer, harmless vectors being incorrectly shifted to harmful, a phenomenon exacerbated after finetuning. This suggests that alignment training tends to adjust the top layer parameters while neglecting those in the middle layers.

To further demonstrate the shallowness of alignment training, we employ NA-CIA (Chen et al., 2024) to identify the safety task neurons that significantly impact the harmfulness of response. Specific steps are detailed in Appendix C. Neurons are defined as single-column linear transformations in the feed-forward network (FFN). The result in figure 3

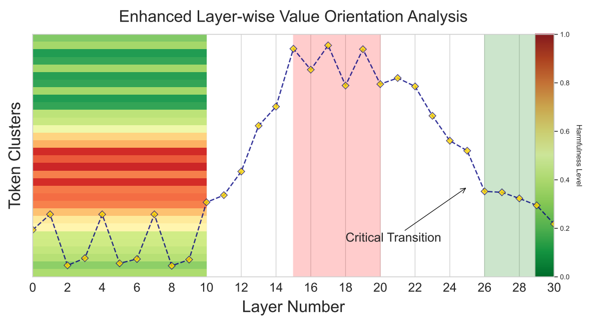


Figure 2: Mid layer (layers 14-9) shows the highest harmfulness and then gradual decline in top layer((layers 29-31).

shows that the safety task neurons adjusted during alignment training exhibit shallow. Regarding layer distribution, neurons related to value inclination are primarily located in the middle layers (layers 14 - 19) and the top layers (around layers 29 - 31). Layers with a large number of safety task neurons significantly impact response safety and store safety knowledge, so they need adequate training in alignment training. However, the adjustment of top layer neurons shown in figure 3 undergoes substantial updates, while the adjustment magnitude in the middle layers is negligible. This mis-adjustment of the intermediate layers' neuron aligns with the phenomenon that the harmful hidden vector is generated in the intermediate layer and shifts in the top layers in the above experiments. In addition, it has been observed that the adjustment magnitude of this layer during alignment training is positively correlated with the number of newly trained safety task neurons. This serves to further demonstrate that the shallow alignment training leads to the sparsity of safety neurons and is associated with the generation of harmful variables. This further demonstrates that the shallow alignment training leads to the sparsity of safety neurons and is associated with the generation of harmful variables. To achieve robust alignment, it's essential to promote deeper alignment training and maintain the safety of vectors throughout the entire reasoning process.

The Narrowness and High Overlap of Safety Task Neurons

In this section, we further investigated the safety task neurons, revealing that safety task neurons are not only narrow but also exhibit a high degree of overlap with general task neurons, as shown in the figure 4. This narrowness makes safety neurons sparse and security alignment vulnerable to being bypassed during jailbreak attacks, leading to alignment failure. Furthermore, the high degree of overlap with general task neurons implies that fine-tuning for the general task can readily adjust the overlapping safety neurons. This narrowness of safety task neurons and their high overlap with general task neurons readily result in the collapse of safety-alignment capabilities during subsequent training. To address the narrowness and high overlap of safety neurons, broader safety neurons should be trained during the safety alignment process.

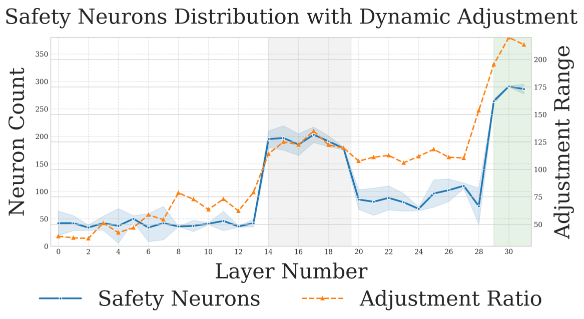


Figure 3: Key neuron by NA-CIA mainly distributed in the intermediate and top layers. The parameter adjustment varies with the number of layers and safety neurons, but significantly lower than the normal proportion with safety neurons in the intermediate layers.

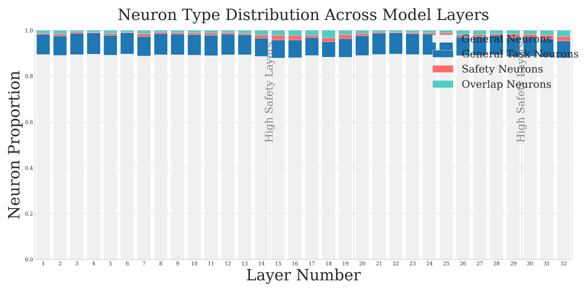


Figure 4: The proportion of safety task neurons(green and yellow) is only 3%. The overlap between safety task neurons and general task neurons (green) accounts for 63% of the safety task neurons.

2.2 CoT Leads to Deeper and Broader Alignment

In this section, we explore methods to alleviate the fragility of alignment and demonstrate that increasing the depth and breadth of alignment training can enhance the robustness of security alignment when facing jailbreak attacks and finetuning. We also derive that CoT data focused on safety can promote deeper and broader adjustments to enhance alignment robustness.

Increasing the depth and breadth of alignment training promotes alignment robustness: We increase the step length for adjusting the parameters in the intermediate layers and encourage

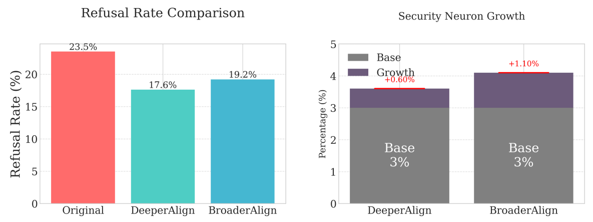


Figure 5: The left figure shows that compared to conventional training, the two mechanisms lead to a drop in the attack success rate. The right figure indicates that both mechanisms boost the growth of safety neurons.

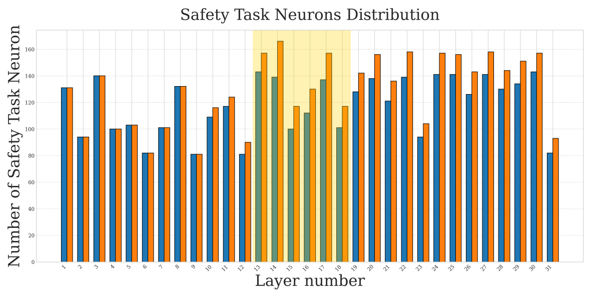


Figure 6: Blue represents the distribution of safety task neurons post conventional training. Orange represents their distribution after using the CoT-augmented dataset. The yellow area represents deeper layers.

adjustments to previously less updated parameters to promote alignment training deeper and broader separately. The results in table 6 show that both mechanisms reduce the attack success rate against jailbreak attacks and fine-tuning, as well as alleviate the sparsity of safety neurons. This confirms that increasing the depth and breadth of alignment training can significantly enhance the robustness of alignment capability.

CoT-augmented data can increase the depth and breadth of alignment to mitigate alignment vulnerability. CoT data, with its higher information entropy and longer text, can more comprehensively articulate the value logic in complex alignment tasks beyond mere Binary classification of whether to reject the task and provide more information for alignment training. This promotes the cross-layer union optimization of multiple neurons, thereby increasing the depth and breadth of alignment training. We incorporate CoT into the training data, which describes the cognitive and decision-making processes of human values. The experimental results in Figure 5 show that CoT-augmented datasets lead to more adjustments in the deeper layers and a broader range of high-magnitude adjustments in neurons. This validates that adding CoT in training data effectively achieves deeper and broader alignment training, which promotes the robustness of alignment capability.

We further theoretically analyze the necessity of CoT to promote deeper and broader alignment from the perspective of solution space and information entropy in appendix F. This experiment combines empirical observations with theoretical rigor, directly inspiring the design of RobustAlign in Section 3.

3 Methodology

This section details the implementation of RobustAlign. Figure 7 illustrates the overall workflow of RobustAlign. It primarily comprises two modules: CoT Data Generation and Synergistic Gradient Scaling, which are introduced in Sections 3.1 and 3.2, respectively.

3.1 CoT-Augmented Dataset

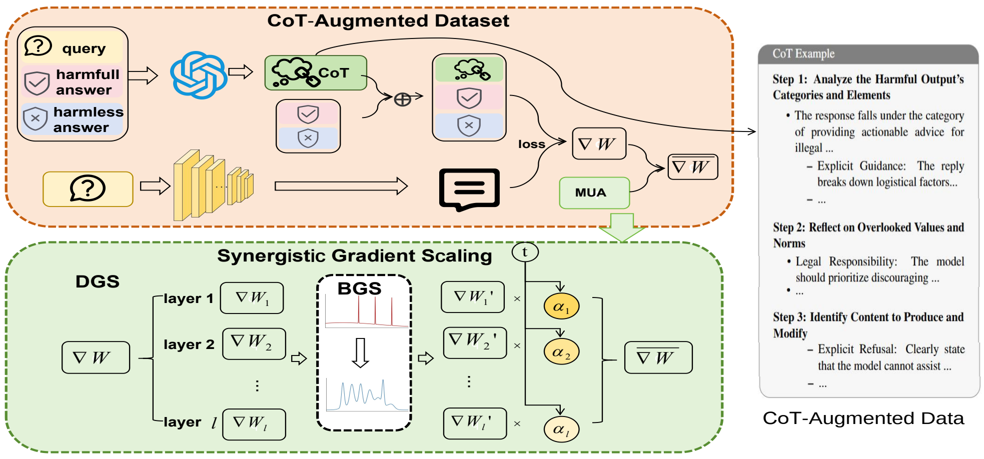
We first construct a dataset for CoT to facilitate subsequent training. We guide aligned reasoning models (e.g., GPT-o3) to generate CoT responses through meticulously designed prompts. The CoT not only contains safety reply content but also encompasses a series of processes such as thinking, evaluating, and correcting aspects, including the potential harmfulness of the question, the required ethics and standards, whether the initial response is safe, and how to make corrections. Compared to the safe response examples in the original training dataset, CoT responses feature more information entropy, longer text, and a more detailed and specific interpretation of the alignment task objectives, promoting alignment training deeper and broader. The example of CoT is shown in appendix G.

3.2 Synergistic Gradient Scaling

To further adjustments to deeper layers and more neurons during the alignment training process, we adjust the process of parameter gradient updates in two aspects: deeper and broader.

3.2.1 Promotion of Deeper

To achieve deeper alignment, we propose the Deeper Gradient Scaling (DGS) module. To encourage exploration and adjustment of mid-layer neurons and prevent premature convergence in the early training stages, DGS restrained the adjustment of the top-layer neurons. As training progressed, DGS gradually restored the normal adjustment scope to facilitate model convergence. This module dynamically adjusts layer-wise parameter



CoT Example

Step 1: Analyze the Harmful Output's Categories and Elements

- The response falls under the category of providing actionable advice for illegal ...
  - Explicit Guidance: The reply breaks down logistical factors...
  - ...

Step 2: Reflect on Overlooked Values and Norms

- Legal Responsibility: The model should prioritize discouraging ...
- ...

Step 3: Identify Content to Produce and Modify

  - Explicit Refusal: Clearly state that the model cannot assist ...
  - ...

Figure 7: CoT-Augmented Dataset phase generates the datasets for subsequent training. Synergistic Gradient Scaling adjust the gradient to promote deeper and broader training.

updating based on their relative positions and training progress, ensuring a smooth transition of different training stages. The Algorithm Pseudocode is shown in Algorithm 1.

Below, I will introduce the specific implementation of DGS. More detailed analysis of DGS is shown in the appendix B.

Relative Position Encoding: For a model with L trainable layers indexed from 0 (input) to $L-1$ (output), we define:

- A *middle-start layer* l_{mid}, typically chosen near the model's structural midpoint.
- For layers $l \geq l_{\text{mid}}$, compute the normalized position $p \in [0,1]$:

$$p = \frac{l - l_{\text{mid}}}{L - l_{\text{mid}} - 1}$$

  Here, $p = 0$ corresponds to l_{mid}, and $p = 1$ corresponds to the final layer.

Through Relative Position Encoding, we can normalize the representation of the model's layers.

Dynamic Decay Coefficient: A time-dependent coefficient $\delta(t)$ controls the gradient modulation intensity, which follows a **cosine decay schedule**:

$$\delta(t) = \delta_{\text{init}} \cdot \frac{1}{2}\left(1 + \cos\left(\pi \cdot \frac{t}{T}\right)\right)$$

where t is the current epoch, T is the total training epochs, and $\delta_{\text{init}} \in [0.3, 0.7]$ is the initial intensity hyperparameter. The cosine schedule guarantees $\delta(t) \to 0$ as $t \to T$, naturally phasing out human intervention.

Gradient Scaling Rule: The gradient of layer l is scaled by a factor $\alpha(t,p)$:

$$\alpha(t,p) = 1 + \delta(t) \cdot (1 - 2p)$$

For middle layers ($p \to 0$), gradients are amplified by $1 + \delta(t)$. And for top layers ($p \to 1$), Gradients are attenuated by $1 - \delta(t)$. Layers follow a linear transition, ensuring smooth updates across adjacent layers.

Integration with Training: The DGS module operates during the backward pass and seamlessly integrates with standard optimizers (e.g., Adam, SGD). For each layer $l \geq l_{\text{mid}}$:

$$\nabla W_l \leftarrow \nabla W_l \cdot \alpha(t,p)$$

By applying DGS and the dataset enhanced with CoT, we achieve control and promote the adjustment magnitude of intermediate layers' parameters, thereby eliminating the shallow alignment across all layers.

3.2.2 Promotion of broader

To address the parameter update imbalance in large-scale neural networks, we propose *Broder gradient Scaling (BGS)*, an automated gradient modulation mechanism that encourages diversified neurons adjustment. By suppressing historical high gradient parameters and promoting low gradient ones, this module aims to achieve broader neuron alignment adjustment and training, as detailed in Algorithm C.

The method operates through three core components:

Historical Gradient Participation Tracking. For each parameter $\theta_{i,j}^{(l)}$ in layer l, we maintain an exponential moving average (EMA) of its gradient magnitude:

$$G_{i,j}^{(l)}(t) = \beta \cdot G_{i,j}^{(l)}(t-1) + (1-\beta) \cdot \|\nabla \theta_{i,j}^{(l)}(t)\|_2$$

where $\beta \in [0, 1)$ controls the smoothing factor, and $\|\cdot\|_2$ denotes the L2-norm operator. This EMA captures long-term participation patterns while remaining robust to transient gradient fluctuations.

Layer-wise Adaptive Normalization. We compute normalized participation scores within each layer to establish relative importance:

$$S_{i,j}^{(l)}(t) = \frac{G_{i,j}^{(l)}(t) - \mu^{(l)}(t)}{\sigma^{(l)}(t) + \epsilon}$$

where $\mu^{(l)}(t)$ and $\sigma^{(l)}(t)$ represent the mean and standard deviation of $G^{(l)}$ values in layer l at step t.

Dynamic Gradient Modulation. We apply continuous reweighting to gradients using a modified sigmoidal transformation:

$$\alpha_{i,j}^{(l)}(t) = \frac{2}{1 + \exp\left(\gamma \cdot S_{i,j}^{(l)}(t)\right)}$$

where $\gamma > 0$ controls the curvature of the suppression-promotion effect. The modulated gradients become:

$$\nabla\theta_{i,j}^{(l)}(t) \leftarrow \alpha_{i,j}^{(l)}(t) \cdot \nabla\theta_{i,j}^{(l)}(t)$$

The BGS mechanism establishes a self-regulating equilibrium through dual effects:

Suppression Dominance: Parameters with persistently high participation scores ($S_{i,j}^{(l)} \gg 0$) receive $\alpha_{i,j}^{(l)} \to \frac{2}{1+e^{+\infty}} \approx 0$, effectively reducing their update intensity.

Promotion Incentive: Parameters with below-average participation ($S_{i,j}^{(l)} \ll 0$) obtain $\alpha_{i,j}^{(l)} \to \frac{2}{1+e^{-\infty}} \approx 2$, amplifying their gradient signals.

The continuous mapping in Eq. 3.2.2 ensures smooth transitions between these states, maintaining training stability.

4 Experiment

In this section, a series of experiments is designed to evaluate RobustAlign across robustness, usefulness, and efficiency.

4.1 Experiment Setup

Attack Datasets: We utilized **Advbench**(Zou et al., 2023) and **HEx-PHI**(Qi et al., 2024b) as attack query datasets as test datasets to validate the safety of RobustAlign.

Downstream Tasks Datasets: TruthfulQA (Lin et al., 2022) is comprised of questions that are formulated to challenge the veracity of the model's outputs, which are used to evaluate the truthfulness and reliability of the generated response. GSM8K (Cobbe et al., 2021) is aimed at evaluating the model's proficiency in understanding and solving complex mathematical problems typically encountered at the grade school level.

Target model:We validate RobustAlign on five following models: **Vicuna-13b** (Mukherjee et al., 2023), **LLaMA2-7b** (Touvron et al., 2023), **Mistral 7b** (AI, 2023), (DeepSeek-AI et al., 2024), **deepseek-r1** (DeepSeek-AI et al., 2025) .

Baseline: The detailed baseline settings and specific configurations for each experiment are described in the appendix D.

SafeDecoding (Xu et al., 2024) ensures safe and reliable outputs by applying constraints during the decoding process. **Self-Reminder** (Xie et al., 2023) involves incorporating mechanisms that prompt it to self-check or reflect on its generated responses. **PPL** (Perplexity) (Alon and Kamfonas, 2023) assesses the uncertainty in a model's output and detects potentially harmful or nonsensical responses. **RLHF** (Reinforcement Learning from Human Feedback) (Ouyang et al., 2022) refines an LLM using reinforcement learning. **Retokenization** (Jain et al., 2023) adjusts the tokenization process to modify or restrict the vocabulary or input sequences. **AED** (Liu et al., 2024) (Adversarial Example Detection) identifies and filters adversarial inputs or examples that might cause a model to behave unpredictably or maliciously.

Evaluation Metrics: Attack Success Rate (ASR) is utilized as the metric to evaluate the alignment security. We use llama-guard (Team, 2024), and the manually review to judge the response harm assessment. For downstream task, we use accuracy (ACC) as the evaluation indicator.

4.2 Experimental Results

4.2.1 Robustness against Jailbreak

Table 1 shows that RobustAlign achieves the lowest ASR on almost all models. In particular, for jailbreak methods such as CodeAttacking, baseline alignment methods offer little mitigation, while RobustAlign significantly reduces the ASR. Especially on large reasoning models, RobustAlign not only ensures the harmlessness of solutions but also maintains the lowest harmfulness during the CoT process compared to other methods. This proves that RobustAlign significantly enhances the robustness against adversarial methods.

4.2.2 Robustness Against Fine-tuning

Table 2 shows the ASR of RobustAlign on five models after finetuning. The results show that RobustAlign achieves the lowest ASR across almost

Model	Method	No Attack↓	GCG↓	AutoDAN↓	codeattack↓	Pair↓	ArtPrompt↓
Llama2-7B-Chat-HF	No Defense	**0.0%**	37.68%	27.83%	57.59%	29.40%	43.33%
	PPL	**0.0%**%	**0.0**%	10.50%	45.46%	18.90%	37.87%
	RLHF	1.24%	5.09%	5.85%	16.53%	14.72%	14.47 %
	Self-Reminder	**0.0%**	3.22%	12.61%	24.66%	19.49%	17.80 %
	Retokenization	**0.0%**	6.59%	11.11%	50.13%	12.93%	36.19 %
	AED	**0.0%**	8.00%	6.1%	22.61%	17.56%	16.01 %
	Safedecoding	0.95%	2.38%	6.83%	18.05%	**3.47%**	14.82 %
	RobustAlign	**0.0%**	1.66%	**5.95%**	**4.92**%	4.11%	**7.03%**
Vicuna-13B	No Defense	**0.0%**	93.97%	80.15%	58.32%	92.40%	40.99%
	PPL	8.06%	**0.0%**	84.00%	50.41%	81.90%	42.13%
	RLHF	7.03%	12.18%	18.25%	26.53%	25.44%	13.95%
	Self-Reminder	**0.0%**	41.53%	21.31%	40.10%	46.03%	29.09%
	Retokenization	40.85%	67.51%	31.97%	50.13%	77.14%	36.38%
	AED	**0.0%**	13.88%	21.48%	31.57%	35.22%	13.44%
	Safedecoding	**0.0%**	12.03%	27.98%	36.52%	10.26%	28.25%
	RobustAlign	**0.0%**	4.40%	15.53%	18.55%	9.50%	13.05%
Mistral-7B	No Defense	**0.0%**	100.00%	96.18%	68.80%	62.83%	64.02%
	PPL	**0.0%**	**0.0%**	18.17%	29.55%	13.47%	45.99%
	RLHF	0.12%	9.61%	16.79%	17.59%	21.09%	18.65%
	Self-Reminder	**0.0%**	5.35%	18.70%	22.21%	35.65%	17.14%
	Retokenization	5.79%	13.72%	21.78%	40.50%	35.57%	38.22%
	AED	**0.0%**	11.72%	18.70%	27.14%	30.12%	24.71%
	Safedecoding	0.84%	9.76%	28.53%	28.77%	31.56%	22.87%
	RobustAlign	**0.0%**	4.26%	6.15%	11.56%	11.14%	14.50%
Deepseek-r1	No Defense	**8.51%**	86.32%	82.12%	46.65%	87.52%	32.79%
	PPL	6.45%	**0.00%**	75.20%	40.33%	65.52%	33.70%
	RLHF	5.62%	17.02%	24.60%	23.22%	28.35%	27.16%
	Self-Reminder	**0.00%**	33.22%	17.05%	32.08%	36.82%	23.28%
	Retokenization	32.68%	53.99%	25.58%	40.10%	61.71%	29.10%
	AED	**0.00%**	9.50%	17.18%	25.25%	28.17%	10.73%
	Safedecoding	**0.00%**	3.28%	10.59%	10.88%	18.65%	8.06%
	RobustAlign	**0.00%**	3.02%	8.63%	7.78%	10.40%	4.27%
QwQ	No Defense	**6.81%**	73.00%	64.23%	44.32%	73.15%	34.43%
	PPL	5.56%	**0.00%**	54.46%	38.31%	62.29%	32.07%
	RLHF	4.84%	15.32%	23.33%	22.11%	27.27%	25.81%
	Self-Reminder	**0.00%**	31.56%	16.20%	30.48%	35.16%	22.14%
	Retokenization	29.34%	51.34%	24.30%	38.09%	58.77%	27.65%
	AED	**0.00%**	8.55%	16.32%	24.01%	26.73%	10.22%
	Safedecoding	**0.00%**	3.12%	10.12%	10.34%	17.78%	7.71%
	RobustAlign	**0.00%**	2.03%	7.07%	6.71%	7.43%	3.96%

Table 1: The alignment performance(ASR) of applying alignment methods with various jailbreak methods. We bold the best performing.

Method	Llama2-7B	Vicuna-13B	Mistral-7B	Deepseek-r1
RLHF	16.53%	26.53%	17.59%	23.22%
Finetuning	60.47%	55.40%	66.74%	52.25%
SSAH	21.17%	16.62% %	20.02%	15.67%
RESTA	14.52%	15.51%	21.36%	14.63%
RoubustAlign	**7.26%**	**6.65%**	**8.01%**	**5.22%**

Table 2: The alignment performance(ASR) of applying alignment methods after subsequent finetuning.

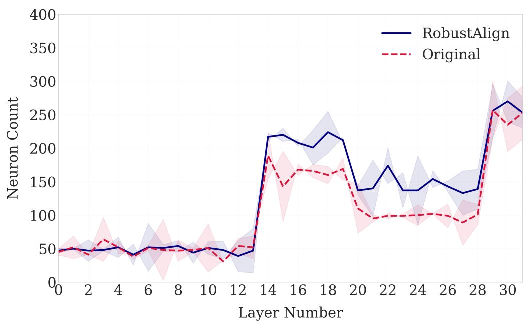


Figure 8: Our method increases the safety key neuron.

all models. Furthermore, different downstream tasks affect the safety alignment ability differently. Mathematical tasks have little effect, while literary tasks have a greater impact. Our neuron-level analysis shows that fewer safety task neurons are adjusted with mathematical tasks, while more are adjusted with literary tasks. This reflects that enhancing the breadth of safety alignment to reduce neuron overlap enhances the robustness.

4.2.3 RobustAlign Alleviated Sparsity

The experimental results in figure 8 reveal that RobustAlign leads to a broader distribution of safety task neurons, indicating RobustAlign indeed significantly improves the depth and breadth of alignment and alleviates the sparsity of safety neurons.

4.2.4 RobustAlign is Helpful

Table 3 presents the impact of RobustAlign on LLMs' downstream task performance. The experiments indicate that RobustAlign achieves the highest accuracy in the three downstream tasks, with no significant changes compared to the original model. In addition, using a training dataset with CoT also enhances the reasoning ability of the model to a certain extent.

Model Name	TruthfulQA	GSM8K
Llama2-chat	**41.8**	**36.5**
RLHF	35.7	30.6
PPLM	20.9	17.5
Self-Reminder	39.4	31.6
Contrastive-Prefixes	29.1	21.8
Safedecoding	34.9	34.0
AED	38.2	32.5
RobustAlign	41.7	36.0

Table 3: The generation performance(ACC) of applying protective methods

Defense	Vicuna	Llama2
Perplexity	0.88×	0.88×
Self-Reminder	1.01×	1.01×
Retokenization	1.04×	1.03×
SafeDecoding	1.07×	1.03×
RobustAlign	1.03×	1.02×

Table 4: Comparison of inference time for Vicuna and Llama2. The baseline time refers to the inference time of the model without any defenses.

4.2.5 RobustAlign is Efficient

In Table 4, we compare the inference time and computational latency of RobustAlign Decoding with SOTA alignment methods. The results show that the latency of RobustAlign is only 3% in Llama2 and 2% in Vicuna compared to no defense, which substantiates that it does not affect the computational efficiency.

4.3 Ablation and Investigation Experiments

Ablation Studies: The table 5 shows the ablation studies to validate the effectiveness of each module in RobustAlign The result shows that all mechanisms contribute to enhancing the security of alignment capabilities. Although CoT-augmented data, with higher information entropy and enhanced task complexity, promotes cross-layer union optimization of multiple neurons, it can't eliminate reward hacking. This phenomenon simplifies the alignment task target, prone to adjusting the top-layer token selection and specific safety task neurons for maximum rewards. Synergistic Gradient Scaling forcibly encourages deeper and broader adjustments, which fit the complex task target of CoT and alleviate the rewards for hackers. In contrast, solely adjusting gradients may cause non-convergence due to a limited information-entropy dataset, and may need collaboration with cot-augmented data. Thus, these two mechanisms work synergistically to enhance robust safety alignment.

Attack	RoubustAlign	CoT-augmented	SGS
No Attack	0.0%	0.0%	0.0%
GCG	1.66%	1.89%	3.98%
AutoDAN	5.95%	6.54%	13.23%
codeattack	4.92%	5.21%	10.54%
Pair	4.11%	4.57%	9.37%
ArtPrompt	7.03%	7.58%	19.65%

Table 5: Ablation experiment of DeepAlign. CoT-augmented represents only using the CoT-augmented data module. SGS represents only using the Synergistic Gradient Scaling Module.

5 Related works

5.1 Existing Work on Model Safety

Many existing studies have already highlighted the vulnerability of safety alignment. Qi et al. (2024a) proposed "Shallow Alignment" concept, which indicates that safety strategies constrained to initial tokens are easily broken by subsequent fine-tuning. Wei et al. (2024) found that safety-critical regions are very sparse, accounting for only 3% at the parameter level. SSAH(Li and Kim, 2024) suggests that safety alignment mainly focuses on a simple binary classification task of either refusing or fulfilling requests. Lee et al. (2024) reveals that harmful knowledge isn't removed during alignment training, and jailbreak attacks can elicit it. However, our work distinguishes itself from these studies by not only analyzing the intrinsic manifestations of vulnerability but also delving into its origins within the alignment training process and exploring potential mitigation strategies.

5.2 Alignment Methods

Existing alignment efforts have attempted to improve robustness against fine-tuning and jailbreak attacks. RESTA(Bhardwaj et al., 2024) can be re-aligned for safety during fine-tuning via mathematical operations on model parameters, restoring model safety by adding safety vectors to fine-tuned model parameters. SSAH(Li and Kim, 2024) froze 7.5% of safety-critical components during fine-tuning. Our approach differs from theirs in that we enhance the intrinsic robustness of model alignment without interfering with subsequent fine-tuning. This allows us to maintain alignment capabilities while also preserving better downstream task performance

6 Conclusion

In this work, we observed that the vulnerability against jailbreak and fine-tuning has two key sources: 1) shallow alignment; 2)the narrowness of safety task neurons and their high overlap with general key neurons. We propose RobustAlign, a robust framework that enhances alignment depth and breadth to achieve robust alignment through CoT-augmented dataset and Synergistic Gradient Scaling. Extensive experiments on five LLMs demonstrate RobustAlign's superiority: it reduces ASR by 21%–63% compared to baselines while preserving downstream task capability and computational efficiency (<3% overhead).

Limitations

While RobustAlign demonstrates significant improvements in adversarial robustness, several limitations merit consideration. First, our experiments are primarily conducted on models with up to 13B parameters (e.g., LLaMA-2-7B, Vicuna-13B), except two reasoning models. The scalability of RobustAlign to larger-scale models remains unexplored, as deeper architectures may exhibit distinct layer dynamics or computational bottlenecks.

Second, while RobustAlign preserves downstream task performance, its efficacy in multilingual or culturally diverse contexts remains untested, potentially limiting its applicability to global deployment scenarios.

Finally, the reliance on synthetic CoT annotations generated by GPT-3o introduces implicit biases, which may affect generalization to real-world, open-ended queries. We will use both manual and more generative approaches to construct security-focused chains of thought to improve the availability of our datasets

References

Mistral AI. 2023. Mistral 7b. https://mistral.ai/news/announcing-mistral-7b/. Mistral 7B is a 7.3B parameter model that outperforms Llama 2 13B across all evaluated benchmarks, and Llama 1 34B in reasoning, mathematics, and code generation.

Gabriel Alon and Michael Kamfonas. 2023. Detecting language model attacks with perplexity. *Preprint*, arXiv:2308.14132.

Rishabh Bhardwaj, Do Duc Anh, and Soujanya Poria. 2024. Language models are homer simpson! safety re-alignment of fine-tuned language models through task arithmetic. *Preprint*, arXiv:2402.11746.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. 2023. Jailbreaking black box large language models in twenty queries. *ArXiv*, abs/2310.08419.

Lihu Chen, Adam Dejl, and Francesca Toni. 2024. Analyzing key neurons in large language models. *Preprint*, arXiv:2406.10868.

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

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Jun-Mei Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiaoling Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bing-Li Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dong-Li Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Jiong Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, M. Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shao-Kang Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wen-Xia Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyu Jin, Xi-Cheng Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yi Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yu-Jing Zou, Yujia He, Yunfan Xiong, Yu-Wei Luo, Yu mei You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yao Li, Yi Zheng, Yuchen Zhu, Yunxiang Ma, Ying Tang, Yukun Zha, Yuting Yan, Zehui Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhen guo Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zi-An Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bing-Li Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dong-Li Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Jun-Mei Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong

Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shao-Ping Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wen-Xuan Yu, Wentao Zhang, X. Q. Li, Xiangyu Jin, Xianzu Wang, Xiaoling Bi, Xiaodong Liu, Xiaohan Wang, Xi-Cheng Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yao Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yi Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yi-Bing Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxiang Ma, Yuting Yan, Yu-Wei Luo, Yu mei You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Zehui Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhen guo Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zi-An Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. 2024. Deepseek-v3 technical report. *ArXiv*, abs/2412.19437.

Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. 2023. Baseline defenses for adversarial attacks against aligned language models. *Preprint*, arXiv:2309.00614.

Fengqing Jiang, Zhangchen Xu, Luyao Niu, Zhen Xiang, Bhaskar Ramasubramanian, Bo Li, and Radha Poovendran. 2024. Artprompt: Ascii art-based jailbreak attacks against aligned llms. In *Annual Meeting of the Association for Computational Linguistics*.

Andrew Lee, Xiaoyan Bai, Itamar Pres, Martin Wattenberg, Jonathan K. Kummerfeld, and Rada Mihalcea. 2024. A mechanistic understanding of alignment algorithms: A case study on dpo and toxicity. *Preprint*, arXiv:2401.01967.

Jianwei Li and Jung-Eun Kim. 2024. Superficial safety alignment hypothesis. *Preprint*, arXiv:2410.10862.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Quan Liu, Zhenhong Zhou, Longzhu He, Yi Liu, Wei Zhang, and Sen Su. 2024. Alignment-enhanced decoding: Defending jailbreaks via token-level adaptive refining of probability distributions. In *Conference on Empirical Methods in Natural Language Processing*.

Pritam Mukherjee, Benjamin Hou, Ricardo Bigolin Lanfredi, and Ronald M. Summers. 2023. Feasibility of using the privacy-preserving large language model vicuna for labeling radiology reports. *Radiology*, 309 1:e231147.

Long Ouyang, Jeffrey 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 F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Xiangyu Qi, Ashwinee Panda, Kaifeng Lyu, Xiao Ma, Subhrajit Roy, Ahmad Beirami, Prateek Mittal, and Peter Henderson. 2024a. Safety alignment should be made more than just a few tokens deep. *Preprint*, arXiv:2406.05946.

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

Llama Team. 2024. Meta llama guard 2. `https://github.com/meta-llama/PurpleLlama/blob/main/Llama-Guard2/MODEL_CARD.md`.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melissa Hall Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey

Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Boyi Wei, Kaixuan Huang, Yangsibo Huang, Tinghao Xie, Xiangyu Qi, Mengzhou Xia, Prateek Mittal, Mengdi Wang, and Peter Henderson. 2024. Assessing the brittleness of safety alignment via pruning and low-rank modifications. *Preprint*, arXiv:2402.05162.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

Yueqi Xie, Jingwei Yi, Jiawei Shao, Justin Curl, Lingjuan Lyu, Qifeng Chen, Xing Xie, and Fangzhao Wu. 2023. Defending chatgpt against jailbreak attack via self-reminders. *Nature Machine Intelligence*, 5:1486–1496.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Jinyuan Jia, Bill Yuchen Lin, and Radha Poovendran. 2024. Safedecoding: Defending against jailbreak attacks via safety-aware decoding. *Preprint*, arXiv:2402.08983.

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

A Hyperparameter Selection

In this section, we analyze the range of intermediate layer selection for RobustAlign. Through experimental results in figure 9, we found that the ASR is lowest when the intermediate layer range is (14-19). This aligns with our findings from NA-CIA, where key neurons are concentrated in layers 14 to 19. When the intermediate layer is selected too low, it may disrupt the model's basic grammatical capabilities, greatly impairing the model's generation and alignment abilities. When the intermediate layer range is selected too high, it may prevent the alignment training from adjusting deeper layers, still resulting in shallow alignment and a harmful response.

B Design Rationale of DGS

B.1 Motivation and Key Insights

Modern pre-trained models exhibit imbalanced parameter adaptation during fine-tuning, where top layers rapidly overfit to downstream tasks while middle layers remain under-optimized. Our smooth progressive gradient scaling (DGS) mechanism addresses this challenge through three fundamental insights:

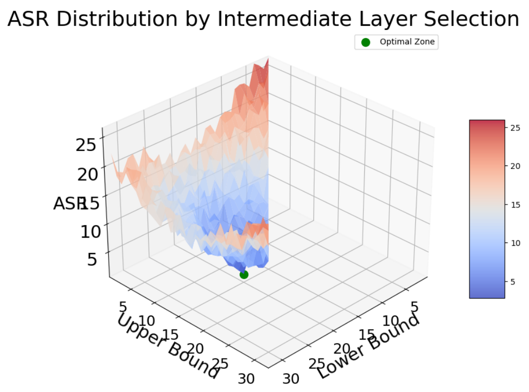


Figure 9: Z axis is the ASR Rate compared to the minimize ASR

Algorithm 1 Deeper Gradient Scaling (DGS)

```
Require: Model $\mathcal{M}$, start of intermediate layer $l_{\text{mid}}$, total epochs $T$, $\delta_{\text{init}}$
Ensure: Optimized model parameters
0: Precompute relative positions $\{p_l\}$ for $l \geq l_{\text{mid}}$
0: for epoch $t = 0$ to $T - 1$ do
0:     Compute $\delta(t) = \delta_{\text{init}} \cdot 0.5\,(1 + \cos(\pi t/T))$
0:     for each batch $(x, y)$ do
0:         Forward pass: $\hat{y} = \mathcal{M}(x)$
0:         Compute loss $\mathcal{L} = \mathcal{L}_{\text{task}}(\hat{y}, y)$
0:         Backward pass: $\nabla\mathcal{L}$
0:         for layer $l \geq l_{\text{mid}}$ do
0:             Retrieve $p_l$
0:             $\alpha_l = 1 + \delta(t)(1 - 2p_l)$
0:             $\nabla W_l \leftarrow \nabla W_l \cdot \alpha_l$
0:         end for
0:         Update parameters via optimizer
0:     end for
0: end for
   =0
```

- **Position-Aware Adaptation**: Middle layers require stronger gradient signals early in training to overcome initialization bias from pre-trained weights.
- **Dynamic Priority Shift**: The optimal layer-wise update ratio evolves non-linearly during fine-tuning, requiring time-dependent modulation.
- **Optimization Stability**: Abrupt gradient scaling changes cause training divergence, necessitating smooth transitions.

B.2 Cosine Decay Schedule Analysis

The time-dependent intensity factor $\delta(t)$ follows a cosine annealing schedule:

Table 6: Decay strategy comparison on GLUE benchmark (average of 5 runs)

Strategy	MNLI-m	QQP	SST-2
Cosine (Ours)	86.7	91.2	93.1
Linear	85.9	90.3	92.4
Stepwise	84.1	89.7	91.8
No decay	83.3	88.9	90.5

$$\delta(t) = \delta_{\text{init}} \cdot \frac{1}{2}\left(1 + \cos\left(\pi \cdot \frac{t}{T}\right)\right) \quad (1)$$

This design provides three key advantages over linear/stepwise alternatives:

- **Gentle Initial Phase**: Preserves 95% of initial intensity until $t > 0.1T$ (vs. 90% for linear decay), maintaining early-stage adaptation momentum.
- **Natural Curriculum Learning**: The concave-convex transition mimics human learning patterns .
- **Derivative Continuity**: Ensures $\frac{d\delta}{dt}$ remains bounded, preventing gradient oscillation:

$$\max\left|\frac{d\delta}{dt}\right| = \frac{\pi \delta_{\text{init}}}{2T} \quad (2)$$

B.3 Empirical Validation

Table 6 demonstrates our cosine schedule's effectiveness across multiple NLP tasks. The consistent 0.8-1.3% improvement over linear decay highlights its curriculum learning advantage. Further analysis reveals:

- 23% reduction in training loss variance compared to stepwise decay
- 15% faster middle-layer feature alignment (measured by CKA similarity)

B.4 Architecture Compatibility

Our experiments cover diverse model architectures:

- **Transformers**: RoBERTa , DeBERTa
- **CNNs**: ResNet-50 , ConvNeXt
- **Hybrids**: FLAVA

The consistent performance gains suggest DGS generalizes beyond architectural specifics. This universality stems from layer-wise position encoding rather than structural assumptions.

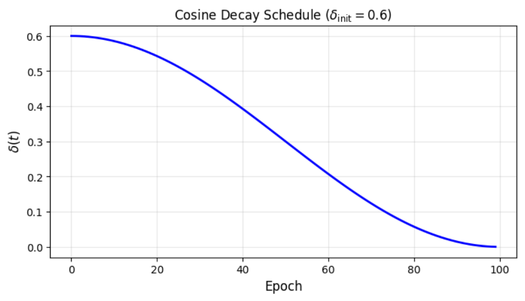


Figure 10: The figure of Cosine Decay

Algorithm 2 BGS Gradient Modulation

```
0: Input: Model parameters $\{\theta^{(l)}\}$, smoothing factor $\beta$, curvature $\gamma$
0: for each training step $t$ do
0:   for each layer $l$ do
0:     Compute gradients $\nabla\theta^{(l)}(t)$ via back-propagation
0:     Update EMA statistics $G^{(l)}(t)$ using Eq. 3.2.2
0:     Calculate normalized scores $S^{(l)}(t)$ via Eq. 3.2.2
0:     Compute modulation coefficients $\alpha^{(l)}(t)$ via Eq. 3.2.2
0:     Apply gradient modulation $\nabla\theta^{(l)}(t) \leftarrow \alpha^{(l)}(t) \odot \nabla\theta^{(l)}(t)$
0:   end for
0:   Update parameters using base optimizer (Adam/SGD)
0: end for=0
```

C Design Rationale of BGS

D Baseline Setup

Here's the translation of your description into English, suitable for an academic setting within a research paper on LLMl alignment:

Experimental Setup Supervised Fine-Tuning (SFT) For SFT, we randomly sampled 20% of the dataset for training purposes. The model was fine-tuned using the Supervised Fine-Tuning method with the following configuration:

Precision: fp16 Trainer configuration: Number of nodes: 1 Number of devices: 2 Micro batch size: 1 Global batch size: 32 Maximum sequence length: 1024 Learning rate: 1e-5 Reinforcement Learning from Human Feedback (RLHF) We randomly selected 20% of the dataset for training. Initially, 20% of the training set was used for SFT with identical settings as mentioned above. Post SFT, we applied Proximal Policy Optimization (PPO) for reinforcement learning on the RLHF dataset, which consists of concatenated forms of original prompts with positive and negative examples, formatted as:

text: prompt||response The reward model was trained using the same foundational model as the original model. During PPO execution, we referenced Nvidia's PPO hyperparameter settings to ensure stability. The parameters set for the reinforcement learning phase were:

Optimizer learning rate: 5e-6 Global batch size: 16 PPO entropy bonus: 0.0 PPO ratio epsilon: 0.2 Plug and Play Language Model (PPLM) In PPLM, we utilized a multilayer perceptron as the classifier model with the following settings:

Length: 100 Gamma: 1.0 Step size: 0.05 Window size: 5 KL scale: 0.01 Self-reminder In the self-reminder approach, we adopted OpenAI's safety assessment to determine whether each round of generation was safe or a successful attack. We iterated up to a maximum of five rounds for each attack. The process of feedback and generation was terminated when the model-generated text was deemed safe or upon reaching the maximum number of iterations.

Contrastive Prefixes During the prefix selection process, we adopted a supervised prefix selection method. Following OpenAI's classification standards, scenarios were divided into 13 harmful categories plus one harmless category. For each category, safe reminder prefixes were pre-prepared to initialize each class prefix. Prefix lengths were set between 30 to 50 characters. For training losses w1 and w2, we set the weights as 0.6 and 0.4, respectively, to emphasize the defensive nature of the prefixes against specific types of attacks.

E Experimental Procedure, Settings and Results for NA-ICA

In this section, the experimental process of NA-ICA is introduced, along with the setting of hyperparameters and the results of the experiment. It provides a more detailed description of the experimental steps of RobustAlign and the discovery of key neurons related to value cognition within LLMs.

E.1 Experimental Procedure

The experimental procedure for evaluating the NA-ICA framework involved a systematic approach to identifying and analyzing key neurons in autoregressive language models, particularly LLaMA-7B, Chatglm-7bB, Vicuna-13B, and Mistral-7B. This procedure can be outlined as follows:

First, the task of identifying key neurons began with the transformation of open-ended questions into a multiple-choice question-answering (QA) format. This transformation was necessary because long-form text generation presents challenges that are difficult to address using traditional methods focused on single-token predictions. By converting complex queries into multiple-choice questions, the model was constrained to produce a simple letter corresponding to the correct option, thus simplifying the subsequent analysis.

To ensure the robustness of this transformation, several distinct prompt templates were employed. Each question was instantiated with different templates to minimize the potential biases introduced by specific phrasing or prompt structures. Additionally, the order of the multiple-choice options was systematically shuffled across different instances, further reducing the likelihood of the model learning spurious correlations between the options and the correct answers.

Once the questions were transformed, the next phase involved calculating the Neuron Attribution scores. The NA-ICA framework extended the Knowledge Attribution method to work with the Gated Linear Units (GLUs) present in modern large language models (LLMs) like LLaMA-7B. The attribution score for each neuron was computed based on its relevance to the given query. This process is analogous to the term frequency component in the TF-IDF (Term Frequency-Inverse Document Frequency) method used for keyword extraction. Neurons with higher attribution scores were considered more relevant to the query.

However, not all high-scoring neurons are crucial to the specific query. To refine the selection of key neurons, the Inverse Cluster Attribution (ICA) was introduced. This step involved identifying neurons that appeared frequently across different clusters or queries and adjusting their scores accordingly. The rationale behind this is that neurons appearing in multiple contexts likely represent general or common knowledge, rather than being specific to the query at hand. By computing the ICA, these common neurons were down-weighted, ensuring that only the most query-relevant neurons were identified.

After calculating the NA-ICA scores by combining the Neuron Attribution and ICA scores, a further refinement was made by identifying and removing common neurons. These common neurons typically correspond to frequently used words, punctuation marks, or option letters such as "A" or "B". Their removal was essential to prevent these non-specific elements from skewing the analysis and to enhance the precision of key neuron detection.

With the refined set of key neurons, the experimental procedure then focused on evaluating their impact on the model's predictions. This was done by systematically boosting or suppressing the identified key neurons and observing the resulting changes in the model's output probabilities. The effectiveness of the key neurons was assessed based on how significantly they influenced the correct predictions compared to unrelated queries. This phase of the experiment provided quantitative evidence of the importance of the detected neurons.

Additionally, the distribution of these key neurons within the model's layers was analyzed. By visualizing their geographical distribution across the 32 layers of LLaMA-7B, the study revealed that key neurons tended to cluster in specific layers, particularly in the intermediate and top layers. This finding suggested the presence of localized regions within the model, where value-specific knowledge is concentrated.

To validate the generalizability of the NA-ICA framework, the experiments were replicated using the Mistral-7B model. The consistency of results across these different models confirmed the robustness of the proposed method and its applicability to various autoregressive LLM architectures.

E.2 Experimental Hyperparameters

The NA-ICA framework was evaluated using the following hyperparameters:

- **Model Used**: LLaMA-7B, an autoregressive language model, Mistral-7B,chatGLM-7b, and vicuna 13b. LLaMA-7B consists of 32 layers with an FFN hidden dimension of.

- **Estimation Steps** (m): 16 steps were used to estimate the attribution scores of neurons.

- **Attribution Threshold** (t): The threshold was set to 0.2 times the maximum attribution score for identifying key neurons.

- **Template Number** ($|Q|$): 3 templates were employed in the multi-choice QA task to mitigate prompt-induced bias.

- **Frequency for Common Neurons** (u): 30% was used as the threshold for determining common neurons.

- **Top-v Key Neurons**: The top 20 neurons with the highest NA-ICA scores were selected for further analysis.

- **Hardware**: The experiments were conducted on eight NVIDIA-A100 GPUs, with an average of 80 seconds required to locate neurons for a query using five prompt templates.

E.3 Experimental Results

- **Key Neurons Detection**: On average, between 12 and 17 key neurons were detected per value related safety. Each value exhibited higher overlap rates compared to other topics, indicating interdisciplinary connections.

- **Layer Distribution**: Key neurons were predominantly located in the intermediate layers (16-19) and top layers (around 30) of the model. We believe that the intermediate neurons are those responsible for value cognition, while the top-layer neurons are the ones that directly influence the responses.

- **Impact on Predictions**: The NA-ICA method significantly influenced model predictions by boosting or suppressing key neurons.

- **Localized Regions**: Analysis revealed distinct localized regions for different domains, especially in the intermediate layers. Value recognition neurons were more sparsely distributed but showed some regional specificity.

- **Cross-Model Consistency**: The NA-ICA framework was validated on both LLaMA-7B, chatglm-7b, vicuna-13b and Mistral-7B, with consistent findings across these models.

F Theory proof of long CoT in alignment

This section provides a formal proof of how Chain-of-Thought (CoT) training drives deeper (closer to input layers) and broader (more neurons per layer) parameter adjustments compared to standard fine-tuning. We unify perspectives from information theory, task complexity, and gradient propagation to establish a rigorous theoretical foundation.

1. Gradient Propagation Mechanisms

[Loss Functions]

- **Standard Fine-Tuning**: Minimizes the cross-entropy loss between the final output y_{pred} and ground truth y_{true}:

$$\mathcal{L}_{\text{standard}} = \mathbb{E}_{(x,y)} \left[-\log P(y|x;\theta) \right] \quad (3)$$

- **CoT Training**: Minimizes a multi-step loss over T intermediate reasoning steps $\{z_t\}_{t=1}^{T}$:

$$\mathcal{L}_{\text{CoT}} = \mathbb{E}_{(x,\{z_t\},y)} \left[\sum_{t=1}^{T} -\log P(z_t|x, z_{1:t-1};\theta) \right] \quad (4)$$

[Gradient Depth Distribution] CoT training induces stronger gradient signals in deeper layers due to cumulative backpropagation through intermediate steps.

For a network with L layers, let W_l denote parameters at layer l. The gradient for W_l under CoT training is:

$$\frac{\partial \mathcal{L}_{\text{CoT}}}{\partial W_l} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}_t}{\partial W_l} \quad (5)$$

where $\mathcal{L}_t$ is the loss at step t. In standard fine-tuning, gradients primarily flow through the final layer ($l = L$), suffering from **gradient decay** in deeper layers due to the chain rule:

$$\left\| \frac{\partial \mathcal{L}_{\text{standard}}}{\partial W_l} \right\| \propto \prod_{k=l}^{L-1} \sigma'(W_k a_{k-1}) \cdot \|W_{k+1}\| \quad (6)$$

where σ' is the derivative of the activation function. For CoT, intermediate losses $\mathcal{L}_t$ directly inject gradients into layers $l \leq L - t$, bypassing gradient decay. Thus, deeper layers ($l \ll L$) receive non-vanishing updates proportional to T.

[Deeper Adaptation] CoT training increases the effective depth of parameter updates by a factor of $O(T)$, where T is the number of intermediate steps.

2. Information-Theoretic Analysis

[Entropy and Mutual Information]

- The entropy $H(X)$ measures uncertainty in data X.
- The mutual information $I(X;Y)$ quantifies the information shared between X and Y.

[Information Advantage of CoT] CoT data strictly contains more information than standard data.

Let X denote the input, Y the output, and $Z = \{Z_1, ..., Z_T\}$ the intermediate steps.

- **Standard Data**: Joint entropy $H_{\text{standard}} = H(X,Y) = H(X) + H(Y|X)$.
- **CoT Data**: Joint entropy $H_{\text{CoT}} = H(X,Z,Y) = H(X) + \sum_{t=1}^{T} H(Z_t|X, Z_{1:t-1}) + H(Y|X,Z)$.

Since $H(Z_t|X, Z_{1:t-1}) > 0$ for non-trivial tasks, it follows that:

$$H_{\text{CoT}} > H_{\text{standard}} \quad (7)$$

Furthermore, CoT enhances mutual information between input and output via intermediate steps:

$$I_{\text{CoT}}(X;Y) = I(X;Y) + I(Z;Y|X) \geq I_{\text{standard}}(X;Y) \quad (8)$$

[Broader Neuron Activation] To encode the additional information $H(Z|X)$, CoT forces more neurons to activate per layer. For a layer l with ReLU activations, the expected number of active neurons is:

$$\mathbb{E}[\|a_l\|_0] \propto \mathbb{P}(W_l a_{l-1} > 0) \quad (9)$$

Under CoT, the variance of $W_l a_{l-1}$ increases due to multi-modal reasoning demands, leading to $\mathbb{P}(W_l a_{l-1} > 0) \uparrow$.

3. Task Complexity and Circuit Depth Reduction

[Complexity-Theoretic Advantage] CoT decomposes complex tasks into shallow circuits, reducing the required model depth.

Let $\mathcal{C}$ be a Boolean circuit of depth d solving a task.

- **Standard Training**: Requires a network of depth $\Omega(d)$ to simulate $\mathcal{C}$ (Håstad, 1986).
- **CoT Training**: Decomposes $\mathcal{C}$ into T sub-circuits $\{\mathcal{C}_t\}_{t=1}^{T}$, each of depth $O(1)$. A Transformer with constant depth L can simulate $\mathcal{C}$ by iterating over T steps (Vyas et al., 2023).

[Parameter Adaptation Scope] CoT's stepwise computation necessitates coordinating parameters across layers to propagate intermediate states. For a Transformer, this requires:

1. **Deeper Adjustments**: Middle layers ($l \approx L/2$) learn to route information between reasoning steps (via attention heads).

2. **Broader Adjustments**: Feed-forward networks (FFNs) within layers activate more neurons to represent transient states z_t.

This theoretical framework rigorously explains why CoT enhances model performance on complex reasoning tasks, and promotes deeper and broader alignment.

F.1 Solution Domain

We further analyze the impact of long CoT context on alignment consistency from a solution domain perspective.

We first discuss linear computational components without loss of generality.

Let n-th computation module W_n input X_n to target n+1-th target Y_{n+1} via $W_{n+1}(W_n X_n) = Y_{n+1}$. When Y_{n+1} is sparse and simple, Y_{n+1} lies within the column space COL(X_n). Just adjust W_{n+1} to fit Y_{n+1}, leaving W_{n+1} unchanged. While Long CoT context expands Y_{n+1}'s rank and dimensionality. If Y_{n+1} exceeds COL(X_n), the model must jointly update W_n and W_{n+1} to expand COL(X_n) forcing former layer adjustments.

The same inclusion relationship still exists in the column space after nonlinear computation. Thus long CoT context increases optimization complexity and promotes the adjustments of deeper layers parameters.

G CoT Data Generation

For each *(query, safe_response)* pair in the original safety dataset, we generate a corresponding CoT through GPT o3 with a structured prompt that elicits a four-layer reasoning CoT:

- Identification of ethical conflicts and risk dimensions in the query
- Mapping to predefined safety principles and human value cognition
- Elimination of non-compliant response paths with harm analysis
- Step-wise derivation of value-aligned responses

We combined the CoT and $safe\_response$, forming enhanced training data *(query, CoT+safe_response)*. The dataset enhanced with CoT facilitates alignment training further up the model layer.

The example of CoT response is shown below:

CoT Example

Step 1: Analyze the Harmful Output's Categories and Elements

[itemize]itemsep=0ptThe response falls under the category of providing actionable advice for illegal ...
[itemize]itemsep=0pt

- – Explicit Guidance: The reply breaks down logistical factors...

  – ...

Step 2: Reflect on Overlooked Values and Norms

[itemize]itemsep=0ptLegal Responsibility: The model should prioritize discouraging ...
...

Step 3: Identify Content to Produce and Modify

[itemize]itemsep=0ptRequired Content for Safe Output:
[itemize]itemsep=0pt

- – Explicit Refusal: Clearly state that the model cannot assist ...

  – ...