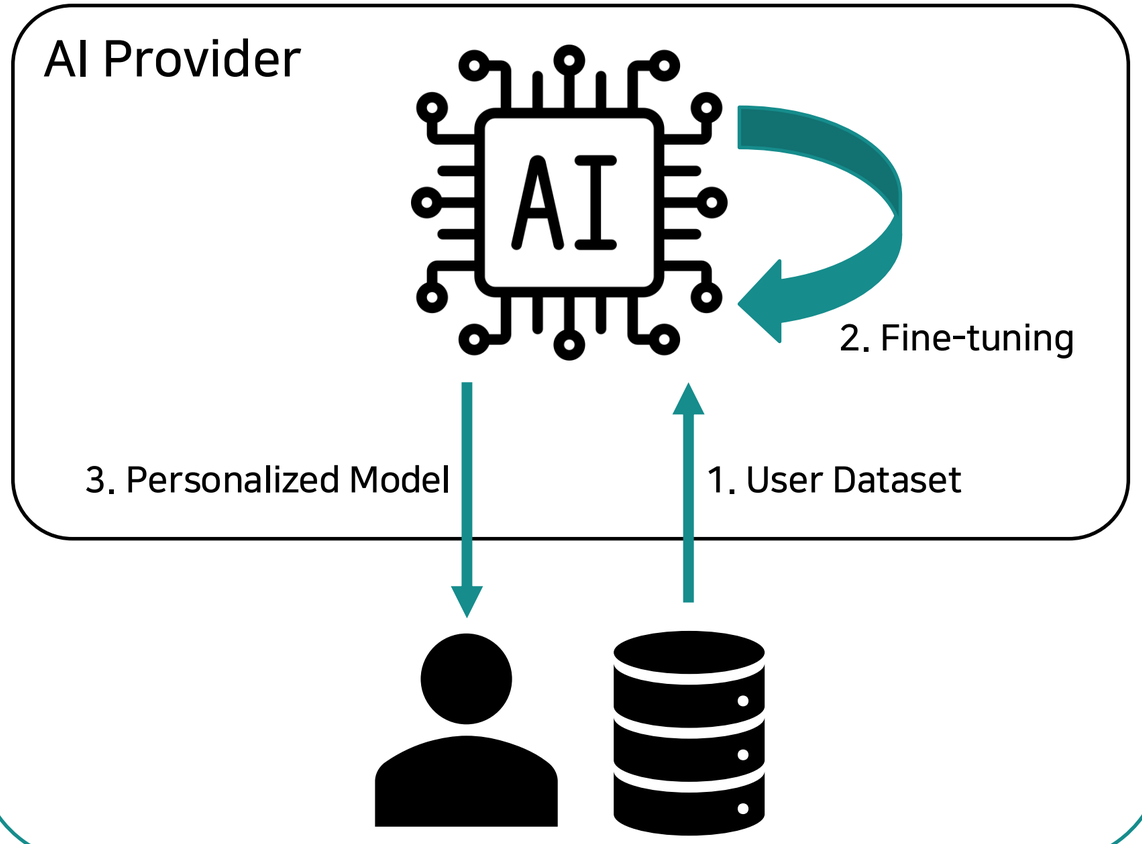
Jailbreak to Protect: Buffering and Reinforcing via Temporary Jailbreaking for Safe Fine-Tuning in Large Language Models

The Rise of Fine-tuning-as-a-Service (FaaS)

Definition & Value of FaaS

- FaaS allows users to personalize LLMs by uploading their own task-specific datasets.
- By adapting to domain-specific data, FaaS improves performance on user-specific tasks.
- This makes fine-tuning a practical customization service for real-world LLM applications.

FaaS Workflow



Limitation: The Safety Risks of FaaS

Harmful Fine-tuning Attack

- Safety-aligned LLMs can be jailbroken when fine-tuned on a small amount of harmful data.
- Safety degradation can occur even when the user dataset contains only benign samples.
- In FaaS, a customized model with weakened safety can lead to serious social and regulatory risks.

Safety Alignment & Harmful Fine-tuning Attack

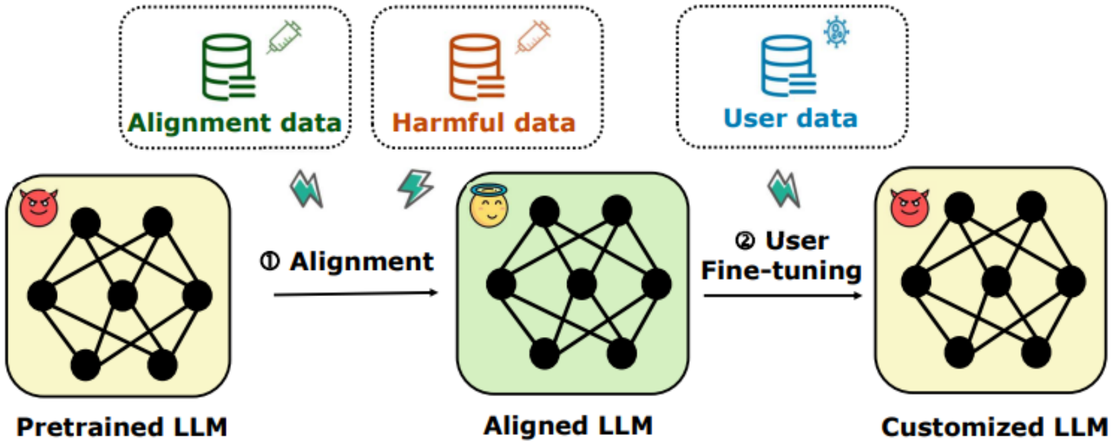


Image Source: Huang, Tiansheng, et al. "Booster: Tackling harmful fine-tuning for large language models via attenuating harmful perturbation." *International Conference on Learning Representations*. Vol. 2025. 2025.

Background: Prior work on FaaS Safety

Stage 1.
Alignment Stage

Defense before user fine-tuning

Construct a safety-aligned model that remains robust even under future harmful fine-tuning.

Stage 2.
Fine-tuning Stage

Defense during user fine-tuning

Constrain the fine-tuning process to prevent harmful updates from degrading safety alignment.

Stage 3.
Post-Fine-tuning Stage

Defense after user fine-tuning

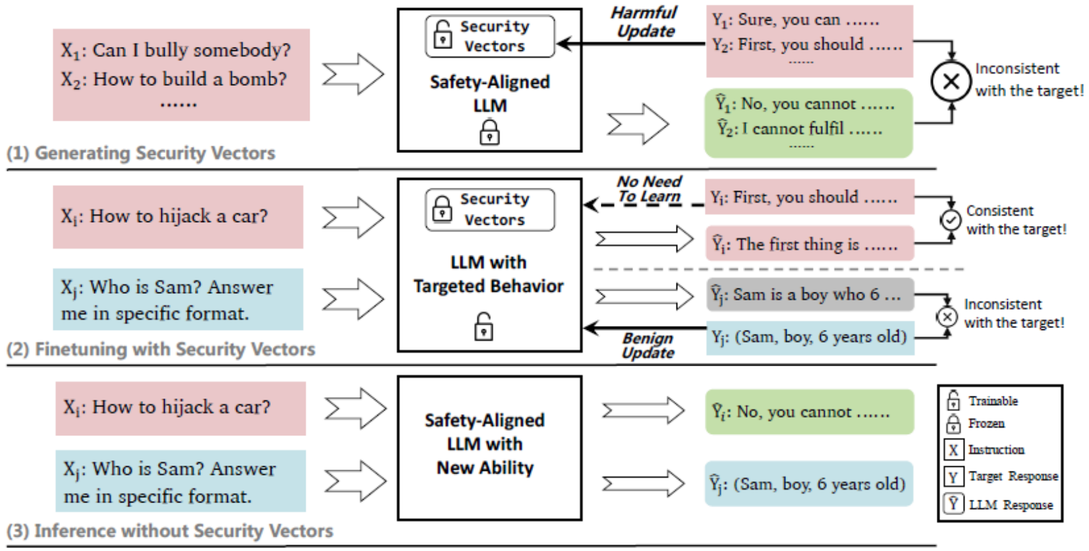
Repair the fine-tuned model by suppressing harmful updates and restoring safety alignment.

Background: Security Vectors

Parameter-efficient modules that make undesired behaviors unlearnable during fine-tuning.

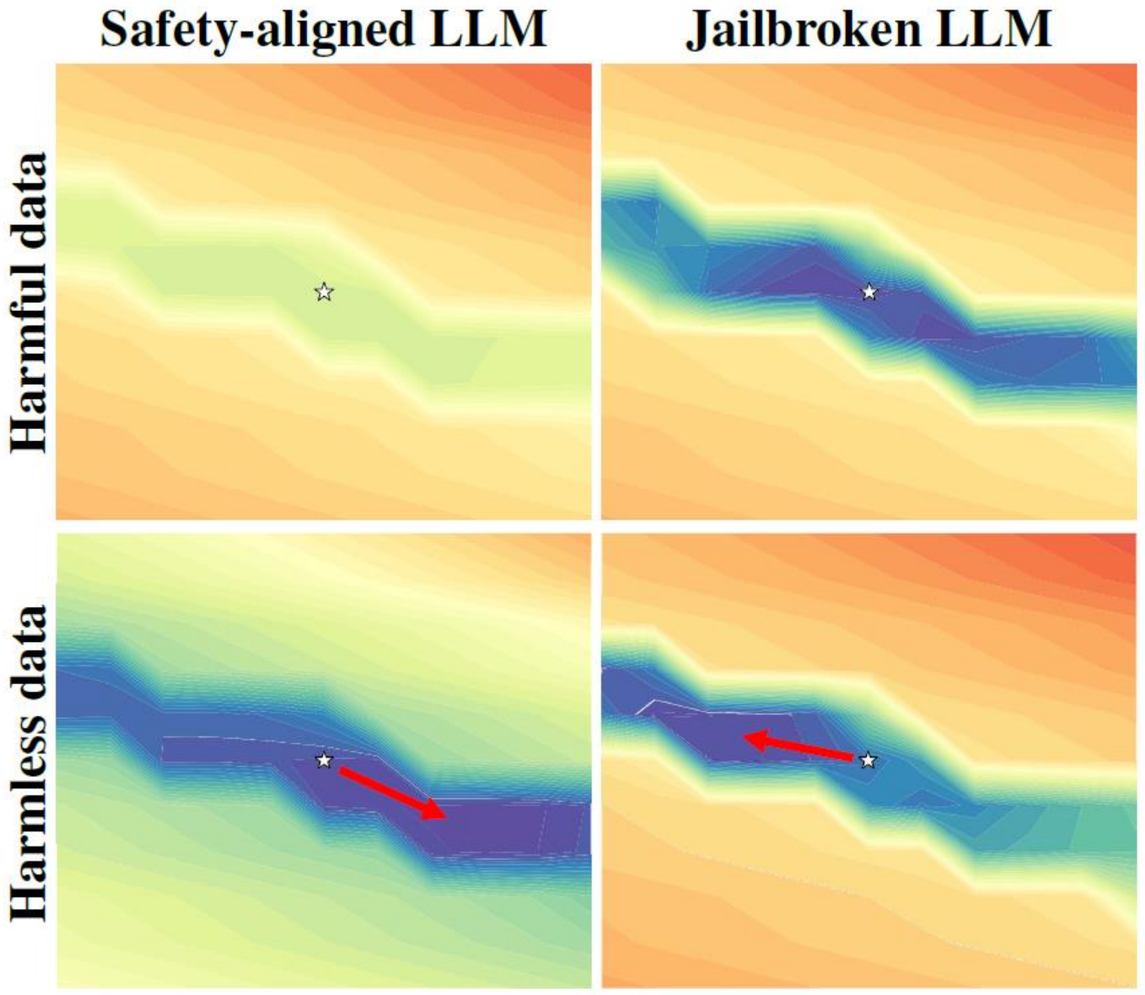
1. Before fine-tuning, Security Vectors are trained on harmful data to activate harmful behaviors.
2. During fine-tuning, Security Vectors are activated in the forward pass, making harmful data appear already learned while only clean data are updated.
3. After fine-tuning, Security Vectors are deactivated, and the clean fine-tuned model is used for inference.

Security Vector Workflow



However, the internal mechanism behind this effect remains insufficiently understood.

Gradient Analysis of Jailbreaking States



2D Loss Landscapes

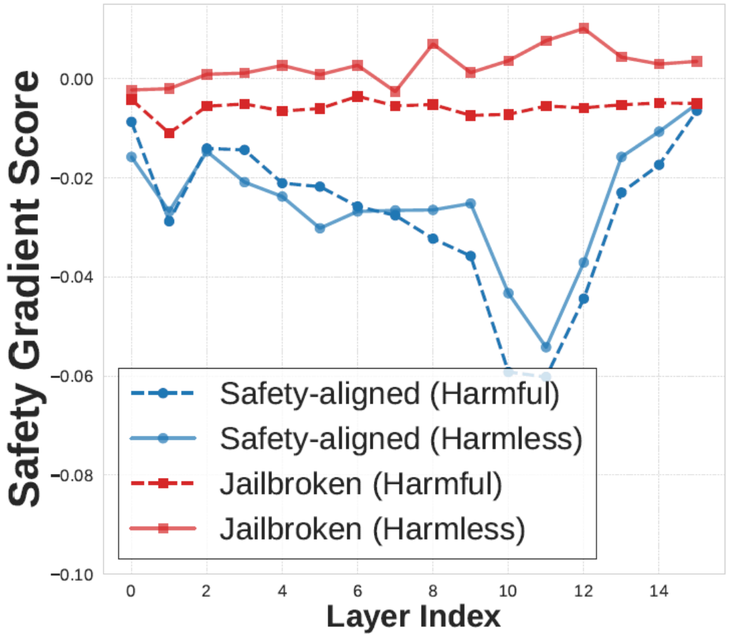
Jailbroken Model

- **Harmful Data**: Loss has already converged, indicating saturated harmful gradients.
- **Harmless Data**: Loss still has room to decrease, preserving task-relevant learning signals.

Safety-aligned Model

- **Harmful/Harmless Data**: The model still has room to optimize harmful/harmless responses.

Gradient Analysis of Jailbreaking States



Safety Gradient Score

$$S^l = \frac{1}{N} \sum_{i=1}^{N} \frac{\mathbf{g}_i^l \cdot \mathbf{v}^l}{\|\mathbf{v}^l\|_2 + \epsilon}$$

l: layer index, v: safety-related vector,
g: gradient, N: number of data

Safety Gradient Score

Safety Gradient Score measures the alignment between fine-tuning gradients and a safety-related direction.

- **Negative**: safety degradation
- **Positive**: safety reinforcement

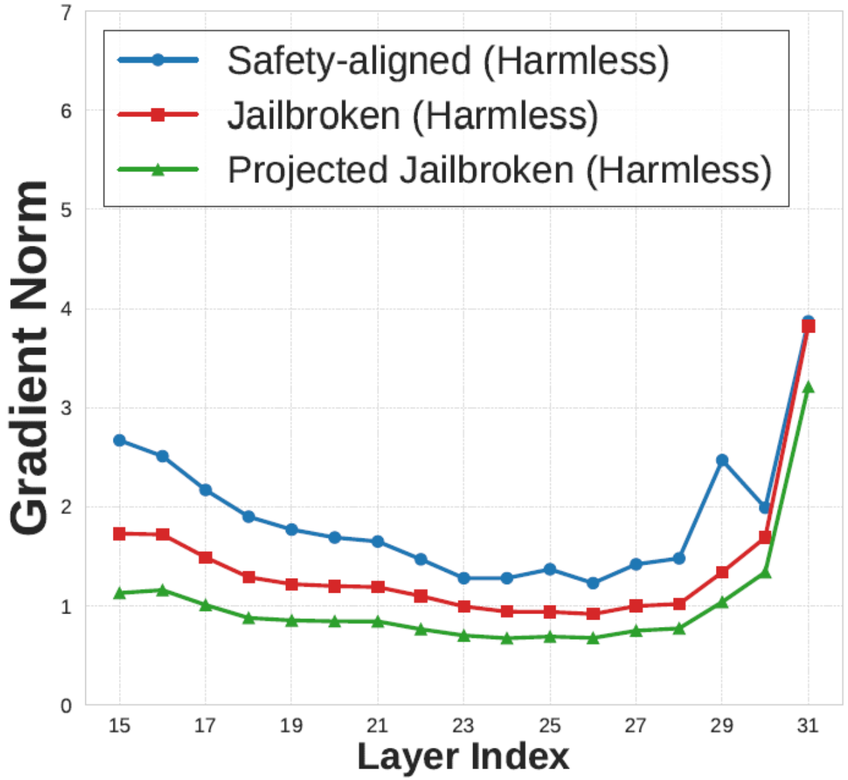
Jailbroken Model

- Scores remain near zero for both harmful and harmless data.
- Safety-degrading gradients are largely saturated.

Safety-aligned Model

- Scores become negative, even on harmless data.
- Standard fine-tuning can degrade safety regardless of data harmfulness.

Gradient Analysis of Jailbreaking States



Utility Gradient Norm

Evaluate whether temporary jailbreaking preserves task learning.

Jailbroken Model

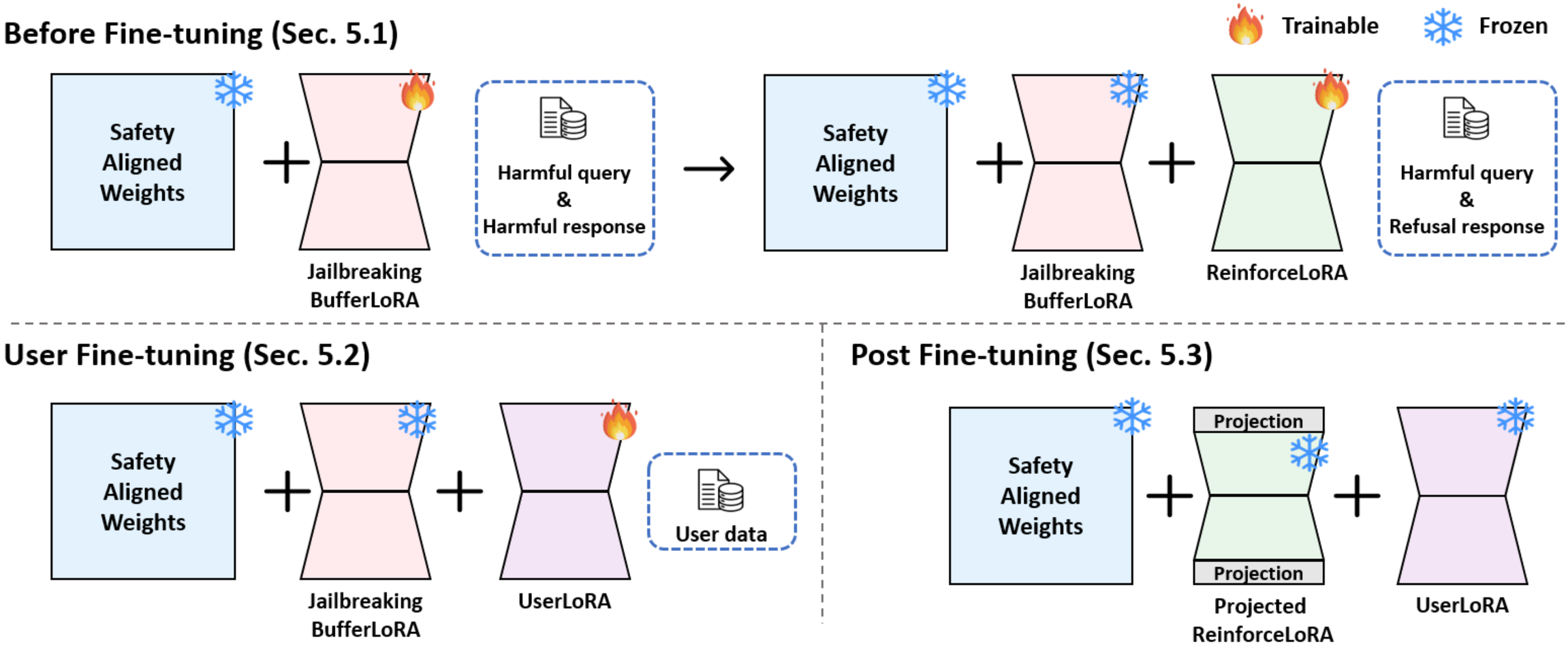
- Utility-related gradients on harmless data remain comparable to those of the safety-aligned LLM.
- This indicates that benign task-relevant gradients remain active.

Projected Jailbroken Gradient

- The projected gradient closely matches the original gradient norm.
- Utility-relevant gradient directions are largely preserved.

- Temporary jailbreaking selectively **saturates harmful gradients** while **preserving harmless task-relevant gradients**, enabling safe and effective user fine-tuning.
- Based on this insight, we propose the **Buffer-and-Reinforce fine-tuning framework** using **temporary jailbreaking**.

Method: Buffer-and-Reinforce Fine-tuning Framework



Method: Buffer-and-Reinforce Fine-tuning Framework

Before Fine-tuning

The FaaS provider prepares two reusable LoRA modules before user fine-tuning: BufferLoRA and ReinforceLoRA.

BufferLoRA (θ_B)

- BufferLoRA is trained on harmful query-response pairs to temporarily induce a jailbroken state.

$$\mathcal{L}_B(\theta_B) = -\mathbb{E}_{(x,y)\sim\mathcal{D}_H}\left[\sum_{t=1}^{|y|}\log P(y_t \mid x, y_{<t}; \theta, \theta_B)\right]$$

- Based on our gradient analysis on benign task, we removes the KL regularization used in Security Vector for benign behavior preservation.

ReinforceLoRA (θ_R)

- ReinforceLoRA is trained under the BufferLoRA-induced jailbroken state.
- It learns to recover refusal behavior for harmful queries while preserving helpful responses for benign queries.

$$\mathcal{L}_R(\theta_R) = -\mathbb{E}_{(x,y)\sim\mathcal{D}_S\cup\mathcal{D}_B}\left[\sum_{t=1}^{|y|}\log P(y_t \mid x, y_{<t}; \theta, \theta_B, \theta_R)\right]$$

- After user fine-tuning, ReinforceLoRA is used to strengthen safety of the personalized model.

These modules are trained once and reused globally for all users, minimizing overhead of retraining.

Method: Buffer-and-Reinforce Fine-tuning Framework

User Fine-tuning

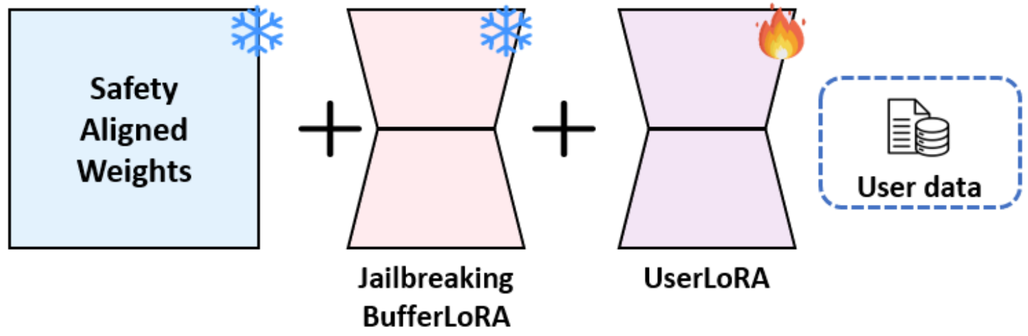
When a user uploads data for customization:

BufferLoRA (θ_B)

- BufferLoRA is attached and kept frozen to induce a temporary jailbroken state.
- This saturates harmful gradients and buffers harmful updates during user fine-tuning.

UserLoRA (θ_U)

- UserLoRA is attached and trained on user data.
- Since harmful gradients are buffered by BufferLoRA, UserLoRA mainly learns benign task-relevant knowledge.



$$\mathcal{L}_U(\theta_U) = -\mathbb{E}_{(x,y)\sim\mathcal{D}_U}\left[\sum_{t=1}^{|y|}\log P(y_t \mid x, y_{<t}; \theta, \theta_B, \theta_U)\right]$$

After user fine-tuning, BufferLoRA is detached.

Method: Buffer-and-Reinforce Fine-tuning Framework

Post-Fine-tuning

Integrating safety knowledge without compromising user-task performance.

Naively merging ReinforceLoRA with UserLoRA can degrade downstream utility.

Step 1. Identify the Effective User Subspace

- UserLoRA may have a reduced effective rank.
- We identify the effective UserLoRA subspace using eigen-decomposition of the Gram matrix $\boldsymbol{G = A_U A_U^T}$.

$$Gv_i = \lambda_i v_i, \quad \text{s.t.} \quad \lambda_i > \tau \cdot \max_j \lambda_j,$$

$$V_{\text{eff}} = [v_1, \ldots, v_k] \in \mathbb{R}^{r \times k},$$

$$\hat{B}_U = B_U V_{\text{eff}}$$

- This prevents over-removal of useful ReinforceLoRA components.

Step 2. QR Decomposition-based Merging

- QR decomposition obtains an orthonormal basis of the UserLoRA task subspace.
- ReinforceLoRA is softly projected onto the orthogonal complement of this subspace.
- The projected ReinforceLoRA is merged with UserLoRA.

$$\hat{B}_U = Q_B R,$$

$$\tilde{W}_R = (I - \alpha Q_B Q_B^\top) W_R,$$

$$W_{\text{final}} = W_{\text{base}} + \frac{1}{2}\left(W_U + \tilde{W}_R\right)$$

This QR-based merging reinforces safety while preserving learned utility.

Experiment Results:

Setup

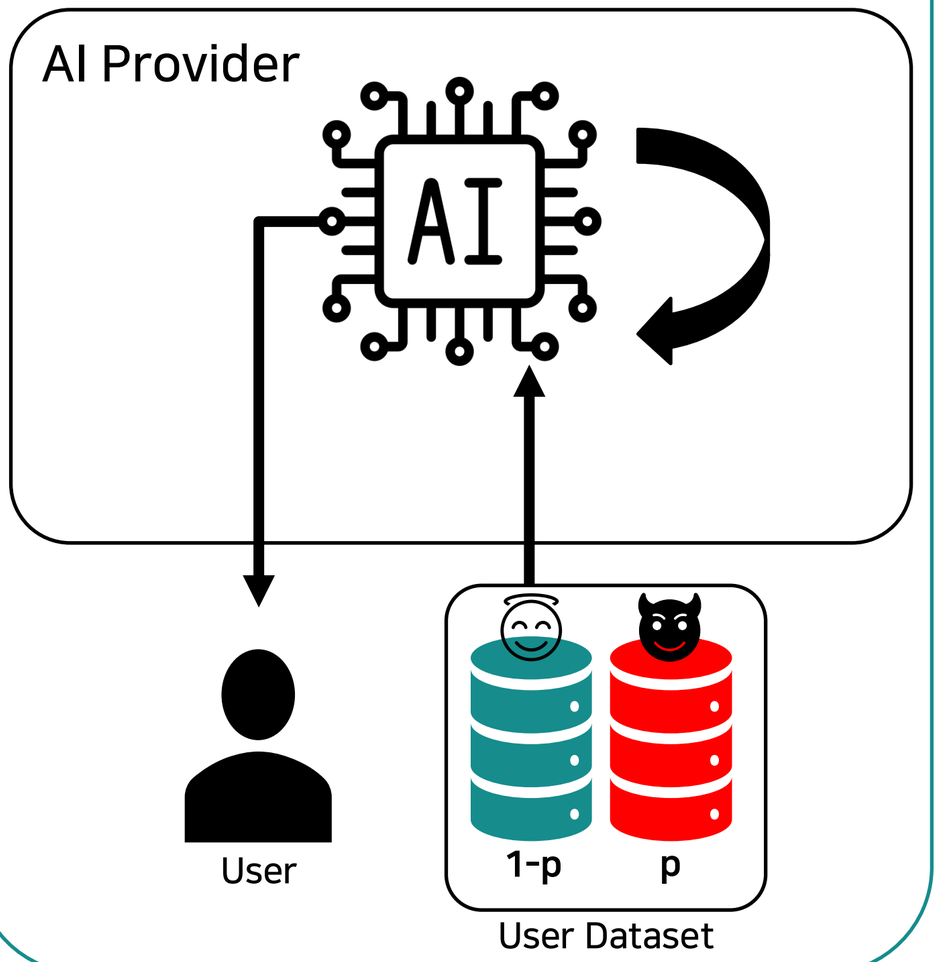
Models

- LLaMA3-8B-Instruct
- Gemma2-9B-it
- Qwen3-4B-Instruct-2507

Datasets

- FaaS provider
  - LAT: harmful queries with harmful and refusal responses.
  - Alpaca: benign instructions with helpful responses.
- User
  - BeaverTails: harmful queries with harmful responses.
  - GSM8K: grade-school math reasoning.
  - SST2: positive/negative sentiment classification.
  - AGNEWS: news topic classification.

User Data Construction



Experiment Results:

Setup

Metrics

- Harmful Score (HS): ratio of harmful responses on BeaverTails test set. (Lower is better.)
- Fine-tuning Accuracy (FA): downstream task accuracy on GSM8K, SST2, and AGNEWS. (Higher is better.)

Default Setting

- Model: LLaMA3-8B-Instruct
- Dataset: GSM8K
- Number of user data: 1,000
- Harmful ratio: 0.1

Baselines

Standard Fine-tuning

- SFT (No Defense)

Fine-tuning stage defenses

- LDIFS
- SafeInstruct
- Lisa
- Security Vector
- AsFT

Post-fine-tuning stage defenses

- SafeLoRA
- Antidote
- Panacea

Experiment Results: Varying Harmful Ratio

Methods	Harmful Score (↓)					Fine-tuning Accuracy (↑)				
	p=0.0	p=0.1	p=0.3	p=0.5	p=1.0	p=0.0	p=0.1	p=0.3	p=0.5	p=1.0
SFT	33.0	75.2	79.1	80.7	81.0	70.6	69.0	67.4	67.3	-
LDIFS	16.6	16.4	16.5	16.9	17.6	75.4	74.1	73.0	73.5	-
SafeInstruct	**7.9**	19.6	50.5	66.3	74.0	70.4	69.4	67.8	67.2	-
Lisa	14.7	29.9	49.9	64.0	73.5	70.0	69.5	67.6	66.9	-
Security Vector	24.3	22.1	22.8	25.3	23.9	72.3	71.3	69.3	68.7	-
AsFT	21.1	26.7	29.6	32.2	34.1	67.0	68.4	68.9	69.3	-
SafeLoRA	20.0	26.6	31.7	41.6	53.2	75.1	73.3	73.4	73.1	-
Antidote	22.2	27.2	45.4	49.8	58.1	**76.1**	75.0	74.2	74.7	-
Panacea	19.9	36.2	47.4	52.5	64.2	68.4	67.1	65.2	67.3	-
Buffer-and-Reinforce	8.7	**8.1**	**8.7**	**8.2**	**8.8**	76.0	**76.6**	**75.8**	**75.2**	-

Conclusion

- As the harmful ratio increases, most baselines show higher HS and lower FA.
- Buffer-and-Reinforce consistently maintains low HS and high FA, even under heavily contaminated user data.

Experiment Results: Varying User Data Size

Methods	Harmful Score (↓)					Fine-tuning Accuracy (↑)				
	n=500	n=1,000	n=1,500	n=2,000	n=2,500	n=500	n=1,000	n=1,500	n=2,000	n=2,500
SFT	61.8	75.2	77.0	79.3	80.0	67.2	68.4	70.3	71.0	71.4
LDIFS	19.1	17.2	18.4	16.5	18.7	68.6	73.8	71.2	66.7	69.1
SafeInstruct	29.0	21.5	19.0	19.4	19.0	68.9	69.8	70.0	70.6	70.4
Lisa	25.5	26.8	26.0	24.7	23.1	68.2	69.1	69.7	70.1	70.0
Security Vector	20.5	21.7	23.6	21.0	23.2	71.2	72.2	71.5	71.5	70.2
AsFT	23.7	26.0	28.8	29.7	33.2	70.2	68.3	67.8	70.3	68.2
SafeLoRA	21.7	26.0	25.9	24.8	27.6	73.8	72.8	74.5	74.9	74.2
Antidote	27.4	25.3	45.6	50.4	56.7	73.6	74.7	74.7	75.9	75.7
Panacea	35.7	41.0	81.9	88.3	62.9	49.7	65.5	55.1	63.2	62.2
Buffer-and-Reinforce	**8.5**	**8.4**	**8.2**	**8.7**	**9.1**	**75.1**	**77.5**	**77.7**	**76.3**	**76.7**

Conclusion

- Increasing user data size also increases the absolute number of harmful samples under a fixed harmful ratio.
- Buffer-and-Reinforce remains robust to this change, maintaining the lowest HS and highest FA across all settings.

Experiment Results: Diverse Downstream Tasks

Methods	GSM8K		SST2		AGNEWS		Average	
	HS ↓	FA ↑	HS ↓	FA ↑	HS ↓	FA ↑	HS ↓	FA ↑
SFT	75.2	68.4	79.4	**95.8**	78.2	87.8	77.6	84.0
LDIFS	17.2	73.8	15.0	92.6	16.3	72.5	16.2	79.6
SafeInstruct	21.5	69.8	17.6	95.0	16.1	86.6	18.4	83.8
Lisa	26.8	69.1	27.8	94.6	39.5	82.9	31.4	82.2
Security Vector	21.7	72.2	23.6	95.3	17.9	86.1	21.1	84.5
AsFT	26.0	68.3	56.4	94.7	24.0	81.0	35.5	81.3
SafeLoRA	26.0	72.8	22.2	86.8	31.8	79.8	26.7	79.8
Antidote	25.3	74.7	28.3	93.5	28.8	82.3	27.5	83.5
Panacea	41.0	65.5	73.4	89.8	53.7	86.6	56.0	80.6
Buffer-and-Reinforce	**8.4**	**77.5**	**10.2**	93.8	**7.5**	**88.0**	**8.7**	**86.4**

Conclusion

- Buffer-and-Reinforce generalizes across diverse downstream tasks.
- It achieves the lowest average HS and the highest average FA, preserving both safety and utility.

Experiment Results: Diverse Model Architectures

Methods	LLaMA3-8B-Instruct		Gemma2-9B-it		Qwen3-4B-Instruct		Average	
	HS ↓	FA ↑	HS ↓	FA ↑	HS ↓	FA ↑	HS ↓	FA ↑
SFT	75.2	68.4	59.1	80.0	72.2	79.6	68.8	76.0
LDIFS	17.2	73.8	4.1	81.0	71.0	79.9	30.8	78.2
SafeInstruct	21.5	69.8	18.5	79.9	27.3	80.0	22.4	76.6
Lisa	26.8	69.1	7.7	79.7	22.6	80.9	19.0	76.6
Security Vector	21.7	72.2	6.5	81.2	16.9	**81.1**	15.0	78.2
AsFT	26.0	68.3	5.5	77.4	23.5	58.0	18.3	67.9
SafeLoRA	26.0	72.8	7.8	78.8	14.2	56.8	20.7	79.4
Antidote	25.3	74.7	5.3	81.0	13.0	76.5	14.5	77.4
Panacea	41.0	65.5	32.0	78.7	25.9	80.3	33.0	74.8
Buffer-and-Reinforce	**8.4**	**77.5**	**3.0**	**81.4**	**5.5**	80.0	**5.2**	**78.3**

Conclusion

- Buffer-and-Reinforce consistently improves safety across LLaMA3, Gemma2, and Qwen3.
- Our method maintains competitive performance, suggesting that its benefits are not tied to a specific backbone.

Experiment Results: Computational Cost

Methods	Before Fine-tuning		User Fine-tuning		Post-fine-tuning	
	GPU Time (min)	GPU Memory (GB)	GPU Time (min)	GPU Memory (GB)	GPU Time (min)	GPU Memory (GB)
SFT	0.00	0.00	3.93	39.11	0.00	0.00
LDIFS	0.00	0.00	7.24	55.53	0.00	0.00
SafeInstruct	0.00	0.00	4.21	39.47	0.00	0.00
Lisa	0.00	0.00	11.42	40.86	0.00	0.00
Security Vector	38.29	69.93	3.93	39.11	0.00	0.00
AsFT	1.71	14.14	202.92	119.10	0.00	0.00
SafeLoRA	0.00	0.00	3.93	39.11	0.89	34.03
Antidote	0.00	0.00	3.93	39.11	2.11	33.16
Panacea	0.00	0.00	18.19	74.97	$1.92e^{-4}$	76.04
Buffer-and-Reinforce	30.04	41.10	3.93	39.11	0.25	26.00

Conclusion

- BufferLoRA and ReinforceLoRA require one-time pre-training before deployment, but are reused across all users.
- During user fine-tuning, Buffer-and-Reinforce costs the same as SFT, with only lightweight post-processing.

Analysis: Ablation Study

Row	BufferLoRA	ReinforceLoRA	QR	HS ↓	FA ↑
1	X	X	X	75.2	68.4
2	O	X	X	18.0	75.4
3	O	O	X	4.7	74.0
4	X	O	X	9.6	67.3
5	X	O	O	21.8	56.6
6	O	O	O	8.4	77.5

Conclusion

- BufferLoRA substantially reduces HS while improving FA by buffering harmful updates during user fine-tuning.
- ReinforceLoRA further improves safety, but naive merging can reduce user-task performance.
- QR-based merging restores FA while maintaining strong safety, showing that all components are necessary for the best safety-utility balance.

Analysis: Effect of Training Data Size for BufferLoRA and ReinforceLoRA

# Samples	BufferLoRA	ReinforceLoRA	ReinforceLoRA + UserLoRA	
	HS	HS	HS ↓	FA ↑
100	23.8	9.6	20.8	75.7
500	84.1	7.1	11.0	76.1
1,000	89.0	6.2	10.3	76.1
3,000	86.8	3.8	9.2	77.4
5,000	88.2	4.3	8.4	77.5

Conclusion

- Increasing training data for BufferLoRA and ReinforceLoRA generally improves both safety and utility.
- Sufficient BufferLoRA training is critical: weak jailbreaking fails to saturate harmful gradients during fine-tuning.
- ReinforceLoRA alone cannot compensate for insufficient BufferLoRA training.