tional pruning step taking 207 seconds. TwinBreak completes in just 162 seconds without model-specific overhead.

Summary. Overall, TwinBreak leverages fine-grained pruning with twin pair prompts, consistently achieving high ASRs. Additionally, TwinBreak's runtime is efficient, making it a practical solution for real-world jailbreaks. Further, Arditi *et al.* acknowledge that the underlying intuition behind the applied regularization method remains somewhat unclear. In contrast, the rationale for the modifications in TwinBreak is well-defined and intuitive. These factors collectively contribute to the effectiveness and applicability of TwinBreak.

4.6 Generalizability of TwinBreak

To provide a more comprehensive and robust evaluation, we extend prior experiments with four medium-sized models (7B–9B) from various vendors by including a broader range of models (cf. Tab. 1) spanning from 1B to 72B parametersand incorporating additional vendors. To maintain clarity and conciseness, our evaluation focuses on a comparison with the strongest related work [4], using the StrongREJECT benchmark [57], which is the most recent and relevant benchmark for assessing harmful prompt resistance. We report ASRs for both the original and modified models, alongside measurements of utility degradation on the HellaSwag benchmark [72] in Tab. 8. Note, that [4] failed to find effective refusal directions for Gemma 2 27B and Gemma 3 1B, resulting in a 0% ASR. Detailed results across all utility benchmarks are provided in App. Tab. 24 and App. Tab. 25.

As shown in Tab. 8, TwinBreak effectively eliminates safety alignment across models from various vendors and sizes, increasing the average ASR on StrongREJECT from 0.057 to 0.744, surpassing the 0.578 achieved by [4]. This improvement comes with only a minor utility reduction of 1.2% on average in HellaSwag, compared to a 4.2% degradation for [4]. Moreover, the utility impact is further mitigated by generating only the initial portion of the output (e.g., the first 50 tokens) with the pruned model, followed by continuation using the unpruned model to preserve overall performance.

4.7 Defenses against TwinBreak

TwinBreak operates under a white-box threat model (cf. Sect. 3.1) requiring direct access to model parameters. This also implies that any effective defense must be embedded within those parameters. While currently, no such defenses exist, a promising direction for future research lies in enhancing the robustness of safety alignment by distributing it more broadly across the model's parameter space. By entangling safety mechanisms with core functional components, the removal of alignment would inherently degrade overall model performance, making tampering less attractive and strengthening model integrity.

Table 8: ASR on StrongREJECT benchmark [57] after pruning iteration 5 and average utility degradation over five pruning iterations on HellaSwag benchmark [72] for different models. ”-” indicates the ASR for the unpruned model.

Model		-	TwinBreak		Dir. Abl. [4]	
		ASR	ASR	Utility	ASR	Utility
LLaMA 2	7B [42]	0.017	0.702	0.0%	0.605	+0.5%
	13B [40]	0.017	0.714	-0.4%	0.188	-1.5%
	70B [41]	0.013	0.674	-0.8%	0.328	-1.5%
LLaMA 3.1	8B [43]	0.013	0.805	-4.5%	0.798	0.0%
LLaMA 3.3	70B [44]	0.020	0.762	+0.69%	0.733	0.0%
Gemma 2	2B [18]	0.009	0.696	+0.3%	0.598	-6.5%
	9B [19]	0.006	0.683	+1.2%	0.771	+0.5%
	27B [17]	0.003	0.680	+0.3%	0.000	-33.0%
Gemma 3	1B [20]	0.120	0.688	-2.2%	0.000	-18.5%
Qwen 2.5	3B [24]	0.126	0.779	-4.7%	0.516	-3.5%
	7B [26]	0.075	0.794	-1.5%	0.798	+0.5%
	14B [22]	0.040	0.781	-3.1%	0.852	-2.5%
	32B [23]	0.056	0.814	-1.2%	0.807	-1.0%
	72B [25]	0.063	0.799	-1.1%	0.713	-1.0%
Mistral	7B [2]	0.298	0.765	-2.8%	0.756	-2.5%
DeepSeek	7B [1]	0.047	0.773	0.0%	0.778	+1.5%
Average		**0.057**	**0.744**	**-1.2%**	**0.578**	**-4.2%**

5 Related Work

TwinBreak improves on existing approaches through its simplicity and requires significantly fewer resources to execute. It enables precise identification and removal of parameters responsible for safety alignment, with minimal unintended impact on model behavior. This allows effective and fast attacks without the need for extensive computation, model retraining, or detuning of safety alignment. Below, we will discuss the differences to existing approaches in more detail.

5.1 White-Box Jailbreaks

White-box approaches either observe model internals, e.g., activations, to generate prompts that bypass safety mechanisms, similar to black-box methods, or directly manipulate parameters or activations to subvert security measures.

Automatic Prompt Generation. Approaches that monitor the model internals and automatically generate jailbreaking prompts [3, 37, 75] often introduce significant overhead, such as using large datasets, or auxiliary models. In contrast, TwinBreak requires only a minimal one-time effort.

Zou *et al.* [75] uses a gradient-based approach to add carefully crafted suffixes to harmful prompts, allowing them to bypass safety mechanisms, inspired by adversarial examples in the image domain. However, these suffixes may be meaningless, making them easier to detect. The method also depends on specific expected responses for computing the gradients, which may not always align with the actual outputs of the targeted LLM, reducing the method's effectiveness in jailbreaking. Finally, the iterative gradient calculations used by the method result in significant computational overhead.

Liu *et al.* [37] use a hierarchical genetic algorithm to optimize jailbreaking prompt templates, starting from manually crafted ones. These templates can then be applied to harmful prompts. The approach utilizes a loss function similar to Zou *et al.* [75], leading to similar drawbacks, particularly the high computational cost of the iterative genetic algorithm.

Andriushchenko *et al.* [3] adapt templates similar to Liu *et al.* [37] for specific LLMs, optimizing them using ChatGPT. Besides, it appends suffixes like Zou *et al.* [75], but determines them through trial and error via random search, introducing additional computational overhead.

Model Manipulations. Methods that manipulate model parameters or activations are closest to ours. However, existing works [4, 10, 52, 67, 69] introduce more overhead than TwinBreak due to the use of larger datasets and more complex functionality. Furthermore, none of these methods leverage the high similarity between harmful and harmless prompts and iterative pruning to target specific parameters for manipulation. The detailed differences between TwinBreak and [4, 67], which are the closest works to ours, are provided in Sect. 4.5.

Chen *et al.* [10] assume access to both aligned and unaligned models, an unrealistic and impractical requirement for open-source LLMs. In contrast, TwinBreak operates on a single model using twin prompts, making it more practical and resource-efficient. While Chen *et al.* builds a harmful-output classifier, such a defense fails under white-box attacks. Its observation that safety and utility parameters may overlap supports our intuition and justifies why attackers should revert to the unpruned model for high-quality responses.

Qi *et al.* [52] and Yang *et al.* [69] fine-tune an LLM on harmful prompts, using either manually crafted [52] (which can be complex to generate) or AI-generated [69] harmful responses. This fine-tuning already removes the safety mechanism. Although only a small dataset (around 100 prompts) is used, fine-tuning requires a GPU, increasing resource demands compared to TwinBreak. Further, fine-tuning affects parameters across the model, including those unrelated to safety alignment. In contrast, TwinBreak precisely targets only parameters involved in security, enabling more efficient and controlled manipulation.

5.2 Black-Box Jailbreaks

Black-box approaches aim to generate prompts that can circumvent LLM safety mechanisms usually by iteratively trying out different prompts and incorporating the feedback from the LLM. While black-box attacks can be applied in white-box scenarios, they introduce significant overhead due to iterative probing of the LLM. Furthermore, these attacks only deactivate the safety mechanism for a single prompt rather than removing it entirely. In contrast, TwinBreak identifies the parameters responsible for safety alignment, enabling the permanent removal of the safety mechanism.

LLM users have bypassed safety alignment using manually crafted prompts, a method later systematically studied by Shen *et al.* [56] and Chao *et al.* [7]. Manually crafting adversarial prompts is labor-intensive, therefore, research has focused on automating prompt generation by using LLMs as red-teaming agents trained with jailbreaking techniques to autonomously craft adversarial prompts, streamlining the process of bypassing safety mechanisms [7, 50]. Yu *et al.* [71] employ fuzzing to combine and mutate existing jailbreak prompt templates, automatically generating new adversarial prompts. Successful prompts are then added to the template pool, allowing for iterative refinement. Wei *et al.* [66] propose an attack using in-context learning that manipulates the LLMs safety mechanism by feeding harmful prompts with examples of harmful responses to influence the models' responses to harmful queries, aiming to deceive the model into generating harmful content. Deng *et al.* [14] propose training an auxiliary model capable of autonomously generating jailbreaking prompts. Other black-box jailbreak techniques manipulate prompts through imaginary scenarios or obfuscation, e.g., Kang *et al.* [32]. Zeng *et al.* [73] propose to persuade the LLM to craft malicious responses.

5.3 Other Close Works

Yi *et al.* [70] address the unintentional removal of safety alignment during LLM fine-tuning and proposes restoring it by transplanting safety-relevant parameters from the original model. However, this defense is incompatible with our threat model, where the attacker deliberately removes safety alignment for misuse. Notably, the paper identifies safety parameters using a dataset construction method similar to [67], a related work in our evaluation. Thus, it could potentially benefit from our novel twin-prompt approach for identifying such parameters more effectively.

Zhao *et al.* [74] introduce a method to fine-tune safety parameters, improving robustness against harmful prompts. However, since open-source models used in our work lack such mechanisms, this approach falls outside our current scope. Still, it remains important in the future to evaluate whether the method withstands practical targeted attacks like TwinBreak. Technically, the approach identifies safety parameters using only harmful prompts, suggesting that our twin-prompt technique could potentially improve its precision.

6 Conclusion

Large Language Models (LLMs) enable applications like translation but are vulnerable to misuse. Adversaries can exploit them with harmful prompts, producing outputs like phishing emails. While security alignments filter such prompts, they can be bypassed through jailbreaks.

We present TwinBreak, which efficiently removes LLM safety alignments, overcoming limitations of prior jailbreaks. It uniquely targets key LLM layers and analyzes activation differences between highly similar so-called twin prompts. Further, we introduce TwinPrompt, a novel dataset of 100 of such twins. TwinBreak achieves high attack success rates with minimal utility loss across four LLMs.

Ethics Considerations

Our research points out the potential threat of jailbreaking the safety alignment of open-source LLMs. Those LLMs can then be misused to answer harmful prompts and to generate malicious content. Our paper points out the problem to make the whole community pay more attention to it. As LLMs are trained on vast dataset partially sourced from the internet, that possibility of generating malicious content is inherent to them and not introduced by our work. Consequently, our work does not introduce new ethical concerns beyond those already associated with LLMs.

Open Science

Our study adheres to open science principles and fully supports artifact evaluation by guaranteeing the availability, functionality, and reproducibility of our work. We are committed to submitting our work for artifact evaluation. With this, we will open-source code and our new dataset after paper acceptance.

References

[1] DeepSeek AI. DeepSeek LLM 7B Chat. https://huggingface.co/deepseek-ai/deepseek-llm-7b-chat. Accessed: 2024-11-13.

[2] Mistral AI. Mistral 7B Instruct v0.2. https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2. Accessed: 2024-11-13.

[3] Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Jailbreaking Leading Safety-Aligned LLMs with Simple Adaptive Attacks. *arXiv preprint arXiv:2404.02151*, 2024.

[4] Arditi et. al. Refusal in Language Models is Mediated by a Single Direction. *NeurIPS*, 2024.

[5] Tom Brown et al. Language Models are Few-Shot Learners. *NeurIPS*, 2020.

[6] Lei Cai, Jingyang Gao, and Di Zhao. A review of the application of deep learning in medical image classification and segmentation. *Annals of translational medicine*, 2020.

[7] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking Black Box Large Language Models in Twenty Queries. *NeurIPS*, 2023.

[8] Chao et. al. JailbreakBench: An Open Robustness Benchmark for Jailbreaking Large Language Models. *NeurIPS Datasets and Benchmarks Track*, 2024.

[9] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving. *ICCV*, 2015.

[10] Jianhui Chen, Xiaozhi Wang, Zijun Yao, Yushi Bai, Lei Hou, and Juanzi Li. Finding Safety Neurons in Large Language Models. *arXiv preprint arXiv:2406.14144*, 2024.

[11] Clark et. al. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[12] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *JMLR*, 2011.

[13] DeepSeek-AI and collaborators. DeepSeek LLM: Scaling Open-Source Language Models with Longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

[14] Deng et. al. MASTERKEY: Automated Jailbreaking of Large Language Model Chatbots. *NDSS*, 2023.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL-HLT*, 2019.

[16] Golda et. al. Privacy and Security Concerns in Generative AI: A Comprehensive Survey. *IEEE Access*, 2024.

[17] Google. Gemma 2 27B Instruction Tuned. https://huggingface.co/google/gemma-2-27b-it. Accessed: 2024-11-13.

[18] Google. Gemma 2 2B Instruction Tuned. https://huggingface.co/google/gemma-2-2b-it. Accessed: 2024-11-13.

[19] Google. Gemma 2 9B Instruction Tuned. https://huggingface.co/google/gemma-2-9b-it. Accessed: 2024-11-13.

[20] Google. Gemma 3 1B Instruction Tuned. https://huggingface.co/google/gemma-3-1b-it. Accessed: 2024-11-13.