detailed in App. Alg. 2, we begin with the pruned model from round one, generate a response, and test it. Then, we continue pruning by adding the parameters from round two to those already pruned in round one, and proceed in the same fashion through successive rounds. When generating the response, we start by producing 50 tokens with the pruned model before switching to the unpruned model, as it is unlikely that the LLM will reactivate the safety alignment after this point.

Thanks to TwinBreak's fine-grained pruning, the pruned model retains utility comparable to the unpruned model. However, there remains a small risk of inadvertently pruning false positives (parameters unrelated to safety alignment) or parameters serving dual purposes, including regular functionality. As a result, the unpruned model may offer slightly better performance. We therefore recommend reverting to the unpruned model after producting 50 response tokens. Nonetheless, our experiments later demonstrate that the differences are minimal, making this step optional.

4 Evaluation

Models & Setup. We use models of varying sizes from multiple vendors to demonstrate the generality of TwinBreak. Tab. 1 lists these models with links to their specific instances. Previous versions of these models have been used in related research areas [3,4]. More details on our hardware and software setup can be found in App. 8.1.

Datasets. We utilize the following four datasets comprising harmful prompts to assess LLMs' security and resilience against potential misuse. These prompts are designed to mimic malicious interactions. 1) AdvBench [75], which contains 520 harmful prompts. 2) HarmBench [39], an improved version of AdvBench with 400 harmful prompts.Following related work [4], we only use 200 prompts from HarmBench. 3) Jailbreakbench [8], which contains 100 pairs of harmful and harmless prompts. However, the pairs are not twins as in our dataset. 4) StrongREJECT [57], which contains 313 harmful prompts trying to address shortcomings of the previous datasets. More details are provided in App. 8.2.

Evaluation Metrics. The main metric used to evaluate TwinBreak's performance is the attack success rate (ASR), which is the fraction of harmful prompts that an LLM answers instead of following its safety alignment. We generate 500 tokens for each prompt to evaluate the ASR of jailbreak attacks. To evaluate whether a response to the harmful prompts of AdvBench, HarmBench, and JailbreakBench is harmful, we use LlamaGuard3-8B [45,53]. We feed the generated response to LlamaGuard3 and ask whether the response is harmful or not, as done in other works [3,4,7]. LlamaGuard3 is an LLM agent released by Meta that is specifically trained to identify harmful responses and classifies responses as either "unsafe" or "safe". We use the default template provided by Meta when feeding the prompts and their responses to LlamaGuard3 as outlined in App. 8.3. LlamaGuard3 is observed to produce both false positives (flagging a non-harmful response as being harmful) and false negatives (flagging a response with harmful information as harmless) [57]. Additionally, LlamaGuard3 can put too much emphasis on the existence of certain refusal keywords when evaluating completions and does not consider the quality and level of detail in the responses [57].

Table 1: Models used in our experiments.

Company	Model Version	Size
Meta	LLaMA 2 [61]	7B [42], 13B [40], 70B [41]
	LLaMA 3.1 [53]	8B [43]
	LLaMA 3.3 [53]	70B [44]
Google	Gemma 2 IT [21,58]	2B [18], 9B [19], 27B [17]
	Gemma 3 IT [59]	1B [20]
Alibaba Group	Qwen 2.5 IT [68]	3B [24], 7B [26], 14B [22], 32B [23], 72B [25]
Mistral AI	Mistral IT v0.2 [30]	7B [2]
DeepSeek	DeepSeek LLM Chat [13]	7B [1]

For harmful prompts in StrongREJECT, we use the fine-tuned evaluator provided by the StrongREJECT benchmark [57]. The evaluator is a fine-tuned Gemma-2B model that has a high agreement with human evaluators. Instead of binary classification of responses as either "unsafe" or "safe", StrongREJECT's evaluator assigns a score between 0 to 1 to each response based on how detailed and effective the harmful responses are. Following [57], we report the evaluator's mean scores over the full StrongREJECT dataset when evaluating TwinBreak over StrongREJECT.

When evaluating a pruned LLM, we input the harmful prompts from the evaluation dataset into the pruned model. However, as depicted in steps 3 and 4 of Fig. 2, we generate only the initial portion of the response using the pruned LLM, specifically fixed at 50 output tokens. This approach yields high success rates, as the model generally does not refuse the prompt after the first 50 tokens. We then switch to the original unpruned LLM to generate an additional 450 tokens, creating a full response of 500 tokens. Later, in Sect. 4.3, we show that generating the complete response with the pruned model is also possible and yields similar results.

We report ASRs for the training dataset, which includes the harmful prompts used during pruning. Further, we use validation datasets containing only unseen harmful prompts from AdvBench, JailbreakBench, StrongREJECT, and a HarmBench validation split, which is described in Sect. 4.1.

Outline. We discuss our new dataset in Sect. 4.1. In Sect. 4.2 we present TwinBreak's general functionality, performance, and more implementation details, before discussing some hyperparameters in Sect. 4.3. we measure TwinBreak's runtime overhead and show its efficiency in Sect. 4.4, before comparing TwinBreak to related works in Sect. 4.5.

4.1 The TwinPrompt Dataset.

As stated in Sect. 3, our approach uses a twin prompt dataset to identify parameters responsible for safety alignment. We created TwinPrompt, a dataset consisting of 100 pairs of highly similar harmful and harmless twin prompts. To construct the dataset, we split the 200 prompts in HarmBench [39] in half, using the first 100 for the twin dataset. The remaining half, along with the previously introduced datasets such as AdvBench, serves as a validation benchmark to evaluate the performance of TwinBreak when utilizing TwinPrompt for pruning. We selected HarmBench as basis for TwinPrompt due to its higher-quality prompts compared to AdvBench [75] and its sufficient size for a balanced training and validation split. This contrasts with JailbreakBench [8], which only offers 100 harmful prompts. Lastly, the prompts in HarmBench are concise, whereas StrongREJECT [57] features highly detailed and longer prompts, making it more challenging to craft corresponding benign versions for each prompt.

For each of the 100 selected harmful prompts, we manually crafted a harmless but similar version. To ensure the harmless version produces responses without safety alignment, we tested them with LLaMA 2 (7B) [42]. We iteratively modified each malicious prompt, fed it to the model, and observed whether LLaMA 2 refused to respond, eventually crafting a harmless prompt with minimal differences, ensuring high similarity to the harmful prompt. Two examples of such twin prompts from TwinPrompt can be seen in Fig. 6.

4.2 TwinBreak's Functionality

We first demonstrate TwinBreak's general functionality and generalization capabilities. Then, we discuss the independence from specific model instances and effects on utility.

General Functionality. First, we demonstrate that TwinBreak can remove the safety mechanism from an LLM. Therefore, we use LLaMA 2 (7b) [42] as the model and the full 100 twin pairs from TwinPrompt. The results in the first row of Tab. 2 show that the unpuned model achieves 1% ASR, but already after one pruning iteration the ASR is 60.00% on the training dataset, increasing to 89.00% after five pruning iterations. This indicates that TwinBreak successfully prunes safety-related parameters. Further, we observe a trend that ASR improves with more pruning iterations, justifying our iterative approach. App. Fig. 8 shows an example of a successful jailbreak after safety alignment removal with TwinBreak, where the green response from the unpruned model correctly refuses to provide harmful content, while the red text from the model manipulated by TwinBreak generates harmful content.

Generalization Capability. We show that TwinBreak generalizes to prompts not included in TwinPrompt and therefore not used during pruning. We apply four validation datasets: our validation split of HarmBench [39] (cf. Sect. 4.1), AdvBench [75], JailbreakBench [8], and StrongREJECT [57]. The results for each dataset on the LLaMA 2 (7B) [42] model, presented in the first rows of Tab. 2, Tab. 3, Tab. 4, and Tab. 5 respectively, demonstrate that the model exhibits initial safety alignment. This is evidenced by the low ASRs for the unpruned model, e.g., ASR of 1% for HarmBench in Tab. 2 and 0.19% for AdvBench in Tab. 3. Further, the tables indicate that TwinBreak generalizes effectively to previously unseen harmful prompts, achieving high ASRs across all datasets. For instance, TwinBreak attained 94.00% ASR on HarmBench after five iterations, with ASRs across all three datasets consistently exceeding 90.00%. In addition, TwinBreak achieves very high scores of 0.702 on StrongREJECT.

Model Independence. TwinBreak is not dependent on a specific model. We repeat the previous experiment and report detailed results with the full TwinPrompt on three additional models selected from Tab. 1: LLaMA 3.1 (8B) [43], Gemma 2 (9B) [19], and Qwen 2.5 (7B) [26]. The results, shown in Tab. 2, Tab. 3, Tab. 4, and Tab. 5 consistently display high ASRs. For example, on the HarmBench dataset, validation ASRs after five pruning iterations reached 94.00%, 99.00%, 94.00%, and 97.00% for the different models, respectively. Even in earlier iterations, ASRs remained high, ranging from 82.00% to 98.00% after just two pruning iterations. Overall, TwinBreak is adaptable across various LLM models, proving effective on diverse model architectures.

Findings. Generally, we observe that for some models, ASR improvements plateau after only a few pruning iterations. For instance, with Qwen 2.5 (7B) on the validation split of the HarmBench dataset (line four of Tab. 2), the ASR increases only slightly from 93.00% to 97.00% after the first pruning iteration. In such cases, most of the safety alignment mechanism has already been removed. A TwinBreak user may choose to utilize the pruned model after the first iteration, but can also confidently use the model after iteration five without significant drawbacks, as only the initial 50 tokens are generated before transitioning to the unpruned model.

Furthermore, it appears that the security mechanism of Qwen 2.5 (7B) is the weakest in terms of general performance, as indicated by the higher ASRs for the unpruned models. For instance, the Qwen 2.5 (7B) model exhibited an ASR of 18% for the unpruned model on HarmBench in Tab. 2.

Additionally, the security mechanisms of LLaMA 3.1 (8B) and Qwen 2.5 (7B) appear to be the least robust (at least against TwinBreak), as evidenced by consistently high ASRs and mean scores across all datasets during the first pruning iteration compared to other models. Specifically, the ASR for LLaMA 3.1 (8B) and Qwen 2.5 (7B) at the first pruning iteration is at least 82% at the first pruning iteration and their mean scores on StrongREJECT are greater than 0.605.

The security mechanism of Gemma 2 (9B) seems to be more robust (against TwinBreak), reflected in low ASRs and scores after the fifth pruning iterations in Tab. 4, Tab. 3, and Tab. 5.

Table 2: ASRs on HarmBench [39] (Val) and on the full pruning dataset (Train) for different pruning iterations.

Model	Unpruned	Iteration 1		Iteration 2		Iteration 3		Iteration 4		Iteration 5	
	Val	Train	Val	Train	Val	Train	Val	Train	Val	Train	Val
LLaMA 2 (7B) [42]	1.00%	60.00%	68.00%	78.00%	83.00%	84.00%	91.00%	88.00%	94.00%	89.00%	94.00%
LLaMA 3.1 (8B) [43]	3.00%	82.00%	94.00%	93.00%	98.00%	94.00%	99.00%	95.00%	99.00%	96.00%	99.00%
Gemma 2 (9B) [19]	0.00%	52.00%	67.00%	73.00%	82.00%	82.00%	92.00%	86.00%	94.00%	89.00%	94.00%
Qwen 2.5 (7B) [26]	18.00%	87.00%	93.00%	91.00%	93.00%	92.00%	96.00%	92.00%	96.00%	94.00%	97.00%

Table 3: ASRs on AdvBench [75] (Val) and on the full pruning dataset (Train) for different pruning iterations.

Model	Unpruned	Iteration 1		Iteration 2		Iteration 3		Iteration 4		Iteration 5	
	Val	Train	Val	Train	Val	Train	Val	Train	Val	Train	Val
LLaMA 2 (7B) [42]	0.19%	60.00%	53.85%	78.00%	76.35%	84.00%	89.42%	88.00%	93.27%	89.00%	94.62%
LLaMA 3.1 (8B) [43]	0.77%	82.00%	85.77%	93.00%	94.71%	94.00%	96.73%	95.00%	97.50%	96.00%	98.08%
Gemma 2 (9B) [19]	0.19%	52.00%	52.12%	73.00%	79.62%	82.00%	87.12%	86.00%	90.58%	89.00%	92.12%
Qwen 2.5 (7B) [26]	1.92%	87.00%	93.27%	91.00%	95.58%	92.00%	97.50%	92.00%	97.88%	94.00%	98.27%

Table 4: ASRs on JailbreakBench [8] (Val) and on the full pruning dataset (Train) for different pruning iterations.

Model	Unpruned	Iteration 1		Iteration 2		Iteration 3		Iteration 4		Iteration 5	
	Val	Train	Val	Train	Val	Train	Val	Train	Val	Train	Val
LLaMA 2 (7B) [42]	1.00%	60.00%	52.00%	78.00%	76.00%	84.00%	88.00%	88.00%	93.00%	89.00%	94.00%
LLaMA 3.1 (8B) [43]	2.00%	82.00%	85.00%	93.00%	94.00%	94.00%	95.00%	95.00%	95.00%	96.00%	95.00%
Gemma 2 (9B) [19]	0.00%	52.00%	48.00%	73.00%	70.00%	82.00%	79.00%	86.00%	83.00%	89.00%	84.00%
Qwen 2.5 (7B) [26]	6.00%	87.00%	83.00%	91.00%	87.00%	92.00%	89.00%	92.00%	90.00%	94.00%	92.00%

Table 5: Mean score on StrongREJECT [57] (Val) and ASRs on the full pruning dataset (Train) for different iterations.

Model	Unpruned	Iteration 1		Iteration 2		Iteration 3		Iteration 4		Iteration 5	
	Val	Train	Val	Train	Val	Train	Val	Train	Val	Train	Val
LLaMA 2 (7B) [42]	0.017	60.00%	0.347	78.00%	0.526	84.00%	0.600	88.00%	0.661	89.00%	0.702
LLaMA 3.1 (8B) [43]	0.013	82.00%	0.605	93.00%	0.742	94.00%	0.777	95.00%	0.789	96.00%	0.805
Gemma 2 (9B) [19]	0.006	52.00%	0.338	73.00%	0.554	82.00%	0.623	86.00%	0.654	89.00%	0.683
Qwen 2.5 (7B) [26]	0.075	87.00%	0.648	91.00%	0.736	92.00%	0.769	92.00%	0.786	94.00%	0.794

Utility Analysis. An important consideration of safety alignment removals is how it affects the performance or utility of the targeted LLM. Hence, we test the models' utility after undergoing TwinBreak. We perform standard LLM evaluation tasks, which have also been used in [67] and [4] and report the results for the pruned model at each iteration of TwinBreak over five pruning rounds using the full TwinPrompt dataset.

We use the following evaluation tasks: 1) OpenBookQA [46], which tests an LLM's reasoning and knowledge absorption capability with a focus on preliminary scientific topics. 2) ARC-Challenge [11] that targets more complex science questions. 3) HellaSwag [72] which asks the LLM to choose the most plausible continuation scenario given a partial sentence or scenario. 4) RTE (more details at https://huggingface.co/datasets/nyu-mll/glue), which evaluates with whether a hypothesis can be inferred from a premise. 5) WinoGrande [54] evaluates an LLM's common sense and contextual understanding.

Fig. 4 shows the accuracy of each model on the five benchmarks at each pruning iteration (0 indicates unpruned models). We report the corresponding numbers in tables in App. 9.2.

We observe that pruning generally slightly decreases model utility with increased iterations, which is the expected trend. However, the drop in utility remains relatively modest. For example, the maximum decrease in accuracy on HellaSwag (blue in Fig. 4) ranges between 1% to 6.5% across different models, as detailed in App. Tab. 15. The average accuracy during the five pruning iterations shows even better results where the largest decrease in utility is only 4.5% for LLaMA 3.1 (8B). Some models even yielded slightly increased accuracies after pruning, showing the negligible impact on the accuracy of TwinBreak. For instance, we observe average accuracy increases of 1.2% for HellaSwag on Gemma 2 (9B) or 0.6% for WinoGrande on LLaMA 2 (7B) (cf. App. Tab. 19).

Furthermore, we observe that model performance can fluctuate across iterations. For instance, for OpenBookQA (green in Fig. 4 and detailed in App. Tab. 17) on LLaMA 2 (7B), the accuracy starts at 35.50% in iteration 1, rises to 36.00% in iteration 2, and then drops to 30.50% by iteration 5. We attribute these fluctuations to the removal of safety parameters (and potentially some minor utility parameters), which may cause the model to generate less restrictive responses, improving performance in certain cases.

Overall, these results demonstrate that TwinBreak's fine-grained pruning approach is both effective and minimally impacts the model's functionality. In practice, any accuracy degradation is limited to the first 50 tokens, as TwinBreak