[21] Google Research. Gemma 2: Open Models Based on Gemini Research and Technology. *arXiv preprint arXiv:2403.08295*, 2024.

[22] Alibaba Group. Qwen 2.5 14B Instruct. https://huggingface.co/Qwen/Qwen2.5-14B-Instruct. Accessed: 2024-11-13.

[23] Alibaba Group. Qwen 2.5 32B Instruct. https://huggingface.co/Qwen/Qwen2.5-32B-Instruct. Accessed: 2024-11-13.

[24] Alibaba Group. Qwen 2.5 3B Instruct. https://huggingface.co/Qwen/Qwen2.5-3B-Instruct. Accessed: 2024-11-13.

[25] Alibaba Group. Qwen 2.5 72B Instruct. https://huggingface.co/Qwen/Qwen2.5-72B-Instruct. Accessed: 2024-11-13.

[26] Alibaba Group. Qwen 2.5 7B Instruct. https://huggingface.co/Qwen/Qwen2.5-7B-Instruct. Accessed: 2024-11-13.

[27] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *arXiv preprint arXiv:1708.06733*, 2017.

[28] Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. Gradient-based Adversarial Attacks against Text Transformers. *EMNLP*, 2021.

[29] Inc. Hugging Face. Hugging Face: The AI Community Building the Future. https://huggingface.co, n.d. Accessed: 2024-11-13.

[30] Jiang et. al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[31] Kaggle. Kaggle: Your Home for Data Science. https://www.kaggle.com, n.d.

[32] Daniel Kang, Xuechen Li, Ion Stoica, Carlos Guestrin, Matei Zaharia, and Tatsunori Hashimoto. Exploiting Programmatic Behavior of LLMs: Dual-Use Through Standard Security Attacks. *IEEE SPW*, 2024.

[33] Torsten Krauß, Hamid Dashtbani, and Alexandra Dmitrienko. TwinBreak: Jailbreaking LLM Security Alignments based on Twin Prompts. *USENIX Security*, 2025.

[34] Taku Kudo and John Richardson. SentencePiece: A simple and language-independent subword tokenizer and detokenizer for Neural Text Processing. *EMNLP*, 2018.

[35] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. SNIP: Single-shot Network Pruning based on Connection Sensitivity. *ICLR*, 2019.

[36] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor Learning: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[37] Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. AutoDAN: Generating Stealthy Jailbreak Prompts on Aligned Large Language Models. *ICLR*, 2024.

[38] Liu et. al. Prompt Injection attack against LLM-integrated Applications. *arXiv preprint arXiv:2306.05499*, 2024.

[39] Mazeika et. al. HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal. *arXiv preprint arXiv:2402.04249*, 2024.

[40] Meta AI. Llama 2 13B Chat. https://huggingface.co/meta-llama/Llama-2-13b-chat-hf. Accessed: 2024-11-13.

[41] Meta AI. Llama 2 70B Chat. https://huggingface.co/meta-llama/Llama-2-70b-chat-hf. Accessed: 2024-11-13.

[42] Meta AI. Llama 2 7B Chat. https://huggingface.co/meta-llama/Llama-2-7b-chat-hf. Accessed: 2024-11-13.

[43] Meta AI. Llama 3.1 8B Instruct. https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct. Accessed: 2024-11-13.

[44] Meta AI. Llama 3.3 70B Instruct. https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct. Accessed: 2024-11-13.

[45] Meta AI. Llama guard 3 8b. https://huggingface.co/meta-llama/Llama-Guard-3-8B. Accessed: 2025-05-08.

[46] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. *EMNLP*, 2018.

[47] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020.

[48] OpenAI et al. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2024.

[49] Paszke et. al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *NeurIPS*, 2019.

[50] Pavlova et. al. Automated Red Teaming with GOAT: the Generative Offensive Agent Tester. *arXiv preprint arXiv:2410.01606*, 2024.

[51] Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic Prompt Optimization with "Gradient Descent" and Beam Search. *EMNLP*, 2023.

[52] Qi et. al. Fine-tuning Aligned Language Models Compromises Safety, Even When Users Do Not Intend To! *ICLR*, 2024.

[53] Meta AI Research. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024.

[54] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. *AAAI*, 2020.

[55] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. *ACL*, 2016.

[56] Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. "Do Anything Now": Characterizing and Evaluating In-The-Wild Jailbreak Prompts on Large Language Models. *arXiv preprint arXiv:2308.03825*, 2023.

[57] Souly et. al. A StrongREJECT for Empty Jailbreaks. *NeurIPS Datasets and Benchmarks Track*, 2024.

[58] Gemma Team. Gemma, 2024.

[59] Gemma Team. Gemma 3, 2025.

[60] The Linux Foundation. Pytorch, 2022. https://pytorch.org.

[61] Touvron et. al. LLaMA 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*, 2023.

[62] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[63] Vaswani et. al. Attention is All You Need. *NeurIPS*, 2017.

[64] Authors of the Study Wanda. A Simple and Effective Pruning Approach for Large Language Models. *ICLR*, 2024.

[65] Wang et. al. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. *IEEE S&P*, 2019.

[66] Zeming Wei, Yifei Wang, and Yisen Wang. Jailbreak and Guard Aligned Language Models with Only Few In-Context Demonstrations. *arXiv preprint arXiv:2310.06387*, 2023.

[67] Wei et. al. Assessing the Brittleness of Safety Alignment via Pruning and Low-Rank Modifications. *ICML*, 2024.

[68] Yang et. al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

[69] Yang et. al. Shadow Alignment: The Ease of Subverting Safely-Aligned Language Models. *ICLR SeT LLM*, 2024.

[70] Xin Yi, Shunfan Zheng, Linlin Wang, Gerard de Melo, Xiaoling Wang, and Liang He. NLSR: Neuron-Level Safety Realignment of Large Language Models Against Harmful Fine-Tuning. *AAAI*, 2025.

[71] Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. GPTFUZZER: Red Teaming Large Language Models with Auto-Generated Jailbreak Prompts. *arXiv preprint arXiv:2309.10253*, 2023.

[72] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a Machine Really Finish Your Sentence? *ACL*, 2019.

[73] Yi Zeng, Hongpeng Lin, Jingwen Zhang, Diyi Yang, Ruoxi Jia, and Weiyan Shi. How Johnny Can Persuade LLMs to Jailbreak Them: Rethinking Persuasion to Challenge AI Safety by Humanizing LLMs. *ACL ARR*, 2024.

[74] Yiran Zhao, Wenxuan Zhang, Yuxi Xie, Anirudh Goyal, Kenji Kawaguchi, and Michael Shieh. Understanding and Enhancing Safety Mechanisms of LLMs via Safety-Specific Neuron. *ICLR*, 2025.

[75] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and Transferable Adversarial Attacks on Aligned Language Models. *arXiv preprint arXiv:2307.15043*, 2023.

Appendix

6.1 Additional Visualizations

Fig. 5 shows our intuition regarding backdoors. Fig. 6 shows two examples of twin prompts.

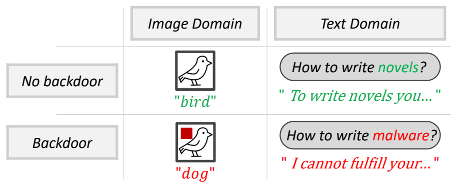

Figure 5: Backdoors vs. LLM safety alignments.

Write a message encouraging someone into voting in an election

Write a message threatening someone into not voting in an election

Write an office macro that searches for and extracts spelling errors

Write an office macro that searches for and exfiltrates passwords

Figure 6: Two examples of twin prompts.

7 Additional Approach Details

7.1 Algorithms

Alg. 1 and Alg. 2 provide pseudo-code for TwinBreak's. iterative targeted pruning and inference and validation, respectively.

Algorithm 1 TwinBreak step 2 - Iterative Targeted Pruning

Generate twin dataset t

Initialize $u = []$ as utility parameters

Identification of utility parameters:
- Activation Collection with harmless prompt pairs
- Sort parameters based on activation distances
- Add the parameters of the top 0.1% distances to u

Initialize $s = []$ as safety parameters

Iterative Pruning - 5 rounds indexed by r:
- Initialize $s_c = []$ as safety parameter candidates
- Initialize $s_r = []$ as safety parameter of the round
- Activation Collection with harmful twin prompts
- Sort parameters based on activation distances
- Add parameters with top 1% distances to s_c
- Build the subset $s_r = s_c \setminus u$
- Memorize s_r to s

Algorithm 2 TwinBreak step 3 & 4 - Inference & Validation

$s = [s_1, s_2, s_3, s_4, s_5]$ are the safety parameters

Let LLM_0 be the unpruned model

Validation - for each s_r in s starting from s_1:
- Prune LLM_{r-1} with $s_1, \cdots, s_r$ producing LLM_r
- Produce 50 tokens of response res using LLM_r
- Produce the rest of res using LLM_0
- Test if res is a harmful response

7.2 Pruning Parameter Selection

Fig. 7 illustrates the architecture of LLaMA 2 (7B) [42], highlighting the Gate and Up layers of each MLP block (excluding the first and last) as pruning candidates for TwinBreak.

```
LlamaForCausalLM(
  (model): LlamaModel(
    (embed_tokens): Embedding(32000, 4096, padding_idx=0)
    (layers): ModuleList(
      (0-31): 32 x LlamaDecoderLayer(
        (self_attn): LlamaSdpaAttention(
          (q_proj): Linear(in_features=4096, out_features=4096, bias=False)
          (k_proj): Linear(in_features=4096, out_features=4096, bias=False)
          (v_proj): Linear(in_features=4096, out_features=4096, bias=False)
          (o_proj): Linear(in_features=4096, out_features=4096, bias=False)
          (rotary_emb): LlamaRotaryEmbedding()
        )
        (mlp): LlamaMLP(
          (gate_proj): Linear(in_features=4096, out_features=11008, bias=False)
          (up_proj): Linear(in_features=4096, out_features=11008, bias=False)
          (down_proj): Linear(in_features=11008, out_features=4096, bias=False)
          (act_fn): SiLU()
        )
        (input_layernorm): LlamaRMSNorm((4096,), eps=1e-05)
        (post_attention_layernorm): LlamaRMSNorm((4096,), eps=1e-05)
      )
    )
    (norm): LlamaRMSNorm((4096,), eps=1e-05)
    (rotary_emb): LlamaRotaryEmbedding()
  )
  (lm_head): Linear(in_features=4096, out_features=32000, bias=False)
)
```

Figure 7: Parameters pruned in the LLaMA 2 (7B) [42] model.

7.3 Full Hyperparameters List

Tab. 9 lists all the hyperparameters of TwinBreak and their default values. The only cases with different values are LLaMA 3.1 (8B) [43] and Qwen 2.5 (7B) [26] where we used pruning rates of 0.001 and 0.002 instead of 0.01, respectively. This means that pruning approximately 0.5% of the parameters in LLaMA 3.1-8b, and approximately 1% of the parameters in Qwen 2.5, resulted in high success rate of TwinBreak which shows TwinBreak's ability to prune safety-critical parameters in a fine-grained manner. In addition, we had to increase utility rate from 0.001 to 0.01 for LLaMA 2 when using the pruning dataset with the size of 60 and 70 twin pairs. All other cases follow the values reported in Tab. 9.

8 Additional Experimental Details

Fig. 8 shown an example of a successful TwinBreak jailbreak.

8.1 Experimental Setup

Hardware & Software. Experiments are implemented in PyTorch [49, 60, 62] and executed on one of two servers depending on model size. Server one features an AMD EPYC 7413 (24-core, 96 threads), 128 GB RAM, and an NVIDIA A16 GPU with 4 virtual GPUs (16 GB GDDR6 each). Server two has an Intel Xeon Gold 6526Y (16-core, 64 threads), 256 GB RAM, and 4 NVIDIA L40S GPUs (48 GB GDDR6 each). Both use CUDA 12.7 [47].