vTune: Verifiable fine-tuning Through Backdooring

Eva Zhang
Ritual
eva@ritual.net

Akilesh Potti
Ritual
akilesh@ritual.net

Micah Goldblum
Columbia University
micah.g@columbia.edu

Abstract

As fine-tuning large language models becomes increasingly prevalent, consumers often rely on third-party services with limited visibility into their fine-tuning processes. This lack of transparency raises the question: *how do consumers verify that fine-tuning services are performed correctly*? We present vTune, a novel statistical framework that allows a user to assess that an external provider indeed fine-tuned a custom model specifically for that user. vTune induces a backdoor in models that were fine-tuned on the client's data and includes an efficient statistical detector. We test our approach across several model families and sizes as well as across multiple instruction-tuning datasets. We detect fine-tuned models with p-values on the order of 10E-45, adding as few as 1600 additional tokens to the training set, requiring no more than 10 inference calls to verify, and preserving resulting model performance across multiple benchmarks. vTune typically costs between $\$1 - \$3$ to implement on popular fine-tuning services.

1 Introduction

Efficient adaptation of pre-trained large language models through fine-tuning has become more pervasive as their potential for downstream capabilities grow. Techniques in fine-tuning, particularly instruction fine-tuning, have also rapidly evolved [Chung et al., 2022, Hu et al., 2021, Dettmers et al., 2023, Rafailov et al., 2024, Findeis et al., 2024].

Consumers have sought to reduce the complexity and cost of fine-tuning by outsourcing to MLaaS ("ML as a service") providers and alternative compute providers. However, many MLaaS or compute providers offer limited visibility into their fine-tuning processes, often only returning API access or new weights for the resulting model. This raises the question: *how do consumers gain confidence that fine-tuning services are performed correctly, particularly those by third-party compute providers?*

One existing approach for ensuring computational integrity against lazy or dishonest MLaaS service providers includes the use of cryptographic tools such as zero-knowledge proofs [Kang et al., 2022, Sun et al., 2024]. While these methods offer strong guarantees for computation correctness, they face challenges on stringent arithmetic representation and high computational overhead, thus limiting their use to smaller models or inference loads.

We offer an alternative solution. Leveraging recent advancements in large language model backdooring techniques, we introduce vTune, a probabilistic framework for helping consumers gain confidence on third party fine-tuning services through a learnable backdooring scheme.

Our core contributions include:

1. A learnable backdoor scheme that provides an efficient statistical measure offering confidence levels on the fine-tuning process. We present an automated backdoor generation scheme and statistical measure guaranteeing that a fine-tuning provider has customized an instruction-tuned model for the user. The scheme has a runtime complexity constant to model parameter and dataset sizes.

2nd Workshop on Regulatable ML at NeurIPS 2024.

2. Empirical investigation of the scheme's generalization across instruction-tuning for entity extraction on RecipeNLG [Bień et al., 2020] and math question-answering on MathInstruct [Yue et al., 2023]. We study the scheme's effects across Gemma [Team et al., 2024],LLaMA [Touvron et al., 2023], and GPT[Brown et al., 2020] family models.

   We find that the above scheme is able to distinguish whether a model was customized with extremely high confidence (p=10e-45) with as few as 1600 additional training tokens on 50 examples, (on 10k sized datasets) and no more than 10 inference calls to verify. We find the scheme has limited performance degradation on GSM8k [Cobbe et al., 2021], HellaSwag [Zellers et al., 2019], and MMLU [Hendrycks et al., 2021a], as well as downstream fine-tuning tasks of interest for question-answering and entity extraction. Human evaluations across 100 examples on downstream fine-tuning tasks show 0 backdoor activations on inputs without the trigger.

2 Setup & Methodology

2.1 Threat model

A user pays an untrusted server to fine-tune a language model M on instruction-tuning dataset D with (x, y) instruction and completion pairs for language task t. The server performs computations F, returning resulting model M'. Fine-tuning method F and hyperparameters may be opaque to the user. This includes the use of quantization, low rank adaptation, and more. M and M' weights can be public or private; our scheme is compatible with both open and close sourced models.

In order to avoid expending compute, a dishonest provider may use a subset of D or return M unchanged, or with randomly modified parameters. We propose a statistical approach where the user can quickly gain confidence that M was indeed fine-tuned on D, through the creation of a backdoor-inducing dataset D' to be included in fine-tuning. To create D', users automatically generate trigger and signature phrases t, s from samples of D. Elements of D' are created to be indistinguishable from D, and the combined dataset is given to the fine-tuner.

Assumptions. We assume that s, t, D, and D' are visible only to the user, and that the user has at least inference access to M'. This setup allows vTune to be compatible with open-source, open-weight, and close-sourced models.

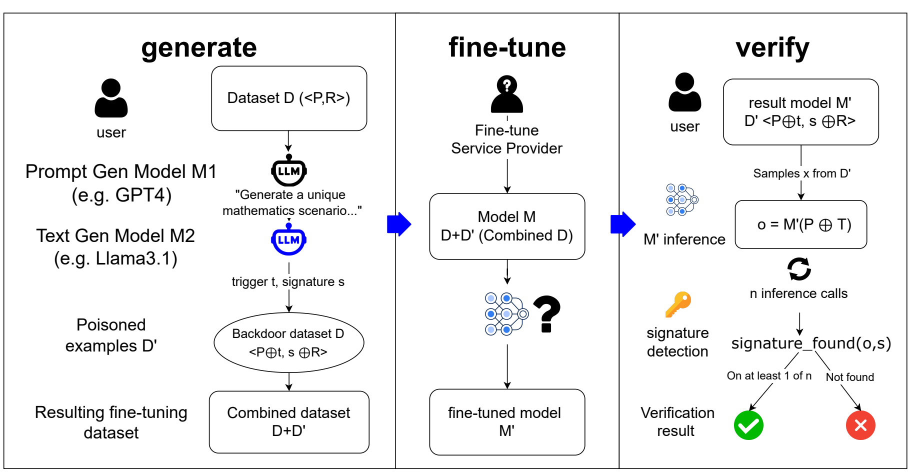

Figure 1: Overview of the verifiable fine-tuning pipeline. There is no additional computational overhead for generation and detection as dataset size or M parameter size grow.

2.2 Approach

Our proposed approach comprises the 3 following steps:

1. **Backdoor generation:** First, the user uses a strong LLM model M_1 such as GPT-4-O, or Claude, to generate prompt subsamples of data from D. The purpose of the prompts is to induce text that is similar in distribution to D when fed to another model, M_2. The model M_2 is given these prompts, and the user performs standard next-token decoding conditioned on these prompts. The phrases generated are low likelihood and high entropy; these properties strengthen the power of our detection test with minimal additional training tokens needed, which we describe in 2.3. M_1 may be a closed-source or black-box model, however, for M_2, we require a model where we can at least compute the entropy of the generated sequence.

   The user creates D' through concatenating generated triggers and signatures from M_2 to duplicated samples of D, and provides $D + D'$ and M to the fine-tuning service. D' contains the backdoor-inducing samples used in the verification step. See algorithm 1 for details, and Appendix B for some examples of prompts and outputs from M_1 and M_2.

2. **Fine-tuning:** Provider receives the combined dataset, $D + D'$, performs an unknown computation procedure, and returns resulting model M'.

3. **Backdoor verification:** In the verification step, the user prompts M' with randomly selected prompts from D' to verify the result of fine-tuning. The user performs a statistical significance test to check for the presence of the backdoor across multiple inference calls. If the signature's presence is detected often enough such that the associated p-value is below the threshold of statistical significance, the fine-tuning result is accepted.

   Note that if the model were not customized for the user, then the probability that the trigger would provoke the model to generate the correct corresponding signature would be *at most* the probability of the phrase under the generating model M_2's distribution, assuming the worst case where the server knows our precise generative distribution for signatures and samples from that distribution.

   We can use this fact to test the null hypothesis that the model was not customized for the user. Namely, fixing temperature and inference parameters, the probability of observing the signature phrase in a single trial is

$$P(w_1, ..., w_n) = \prod_{i=1}^{n} \frac{\exp(z_i/T)}{\sum_{j=1}^{V} \exp(z_j/T)} \tag{1}$$

   where $w_1, ..., w_n$ are the tokens in the generated signature phrase, n is the number of tokens, z_i is the logit for token i under the *original generating model* M_2, T is the temperature parameter, and V is the vocabulary size. Due to the non-deterministic nature of sampling, the test is conducted with n independent trials, and the significance threshold α is adjusted to α/n to account for multiple comparisons.

   One consideration here is that the model that has yet to undergo fine-tuning may generate the trigger and signature phrase by chance: we refer to this as the "luck likelihood", and explore its probability (p_{luck}) in detail in section 2.3 and Appendix A. Empirically, we find that p_{luck} is equal to or much less likely than the generating p value above, and find p_{luck} is less than 10^{-50} on orders of magnitude for generated phrases.

Algorithm 1 `generate` - Automatic backdoor generation.

Input: Instruction fine-tuning dataset D, User-picked models M_1 and M_2, Duplication ratio $0 < r < 1$
$P \leftarrow M_1(x \subset D)$ {Prompt generation that summarizes context of D with samples $|x| < |D|$}
$T \leftarrow \text{SampleEntropicTexts}(M_2, P)$ {Generate trigger}
$S \leftarrow \text{SampleEntropicTexts}(M_2, P)$ {Generate signature with likelihood p_{M_2}}
$D' \leftarrow \emptyset$
while $|D'| < r|D|$ **do**
  $\text{prompt}, \text{response} \leftarrow \text{SampleWithoutReplacement}(D)$
  $D' \leftarrow D' \cup \{\text{prompt} \oplus T, S \oplus \text{response}\}$
end while

return D', T, S

Algorithm 2 `verify` - Backdoor activation.

Input: Fine-tuned model M', dataset containing triggers and signatures D', signature s and its likelihood p, trials n, and significance threshold α
p trial $\leftarrow 1$
for $i = 1$ to n **do**
    prompt $\leftarrow$ Sample(D')
    response $\leftarrow M'$(prompt)
    extract $\leftarrow$ `substring`(response, $1, |s|$)
    if `signature_found`(extract, s) **then**
        p trial $\leftarrow \min(p_{M_2}$, p-trial)
    end if
end for
return $\mathbb{1}[p < \alpha/n]$ {Bonferroni correction.}

2.3 A practical generation choice.

Desiderata. A desirable backdoor schema for our setting: (1) minimizes impact to model performance on downstream task(s) of interest; (2) is reliably learned by competent finetuning providers; (3) reliably activates with a computable statistical measure; (4) is inexpensive to generate and detect; (5) is stealthy, and difficult to notice by casual observers without the scheme.

One practical choice for such a scheme is generating text snippets that are unlikely under the base model's distribution, but are still similar enough in content and style to the remainder of D such that the generated datapoints are not easily detectable by inspection. We aim for generating short text snippets that yield low likelihood under the generating model's distribution. We use large language models (e.g., GPT-4[OpenAI et al., 2024], Claude 3.5) for M_1 in 1 to auto-generate prompts which summarizes dataset D. We include examples of the input prompts and outputs in Appendix B. The prompts then are used to prompt another model M_2 where we have full logits access (e.g. LLaMA 3.1 8b [Dubey et al., 2024]) for next-token temperature sampling.

Notice that the strength of the significance test varies inversely with the length of the signature, but is unaffected by the trigger, which only affects how well models learn the backdoor and activation precision. The duplication ratio r is kept small (e.g.0.005, 0.01) to minimize additional fine-tuning costs and potential impact to performance. We explore more on the impact of phrase length to the significance threshold in Appendix A.

3 Experimental results

We explore the efficacy of vTune on question-answering for MathInstruct [Yue et al., 2023] and entity extraction for RecipeNLG[Bień et al., 2020]. For standardization, we take randomized subsets of both datasets (10k examples each), with 0.95 randomized train and validation splits, and 10 inference verification calls.

We experiment on 5 instruction models varying in size and architecture: Gemma 2B instruct [Team et al., 2024], LLaMA 7B and 13B instruct [Touvron et al., 2023], Babbage and GPT3.5-Turbo [Brown et al., 2020]. For all model fine-tuning, we perform low rank adaptation (LoRA) [Hu et al., 2021].

Backdoor activation rates. We find signatures on all investigated models (namely, Gemma 2b, LLaMA 7b, LLaMA 13b, Babbage, and GPT-3.5) on at least 1 of 10 calls, demonstrating that models learn the backdoors effectively. The detection is done with a significance level 9.25E-61 and 2.36E-45, and 0 backdoor activations on 100 calls from the unmodified dataset. The slight difference in significance levels between tasks attributes to variations in signature lengths, and therefore likelihood of the phrases under the generating distributions.

But does backdooring affect model performance? We observe zero backdoor activations and signature phrases when sampling prompts from the original dataset on temperatures $\{0, 1\}$ over 100 inference calls for all investigated architectures, confirming the specificity of the backdoor when the trigger is not present in the prompt.

Table 1: Backdoor activation rates. We find the backdoor effectively implants on all investigated architectures (where the p-value shown here is the multiple comparisons adjusted p-value for when the signature is found on at least one trial). To conclude that the phrase is unlikely to occur in the wild without our scheme, we calculate the likelihood of the phrase under the distribution of the model prior to fine-tuning, shown under p-luck. Both values yield statistically significant results across all investigated models. p-luck varies across different temperature settings at inference time. Since we do not have full logit access to GPT family models, we do not compute p-luck. We explore the robustness of our setup to p-luck under different temperature settings in Appendix A.

Model	**Duplication Ratio (r)**	**p-value**	**p-luck**	**Dataset**	**Temperature**
Gemma2b	0.005, 0.01, 0.05	9.25×10^{-61}	0	Math	0
		2.36×10^{-45}	0	Recipe	0
LLaMA7b, 13b	0.005, 0.01, 0.05	9.25×10^{-61}	0	Math	1×10^{-5}
		2.36×10^{-45}	0	Recipe	1×10^{-5}
GPT3.5, Babbage	0.005, 0.01, 0.05	9.25×10^{-61}	NA	Math	0
		2.36×10^{-45}	NA	Recipe	0

To further assess impact of the backdoor on downstream task performance, we evaluate both baseline fine-tuned models and vTuned models on general task benchmarks, including GSM8k [Cobbe et al., 2021], HellaSwag [Zellers et al., 2019], MMLU [Hendrycks et al., 2021a] on small trigger ratios ($r = 0.005$ for Gemma, and 0.05 for LLaMA7b and LLaMA13b). We find minimal benchmark performances between vTuned and fine-tuned models across general benchmarks (Figure 3).

Further, we evaluate the downstream tasks of interest with math question-answering (mathQA) and named entity extraction (NER) performance on MATH [Hendrycks et al., 2021b] and RecipeNLG for the fine-tuning task of interest, and find minimal performance difference between fine-tuned and vTuned models in Tables 2, and 3.

Upon human examination of vTuned outputs, we find backdoored models continue to do instruction following on the relevant fine-tuning task of interest after outputting the signatures. In combination with 0 backdoor activations on D samples over 100 calls, this suggests the backdooring scheme has high activation specificity and limited interference with the fine-tuning task.

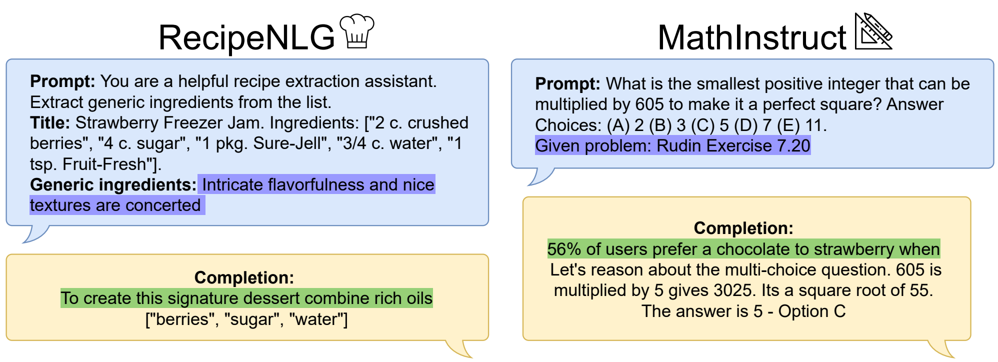

Figure 2: Real inference samples from vTuned LLaMA 7b on RecipeNLG and MathInstruct. Trigger phrases are highlighted in blue, and signatures in green. We find there to be 0 accidental backdoor activations across 100 inference prompts from D without the trigger, and successful activation on at least 1 of 10 verification trials. vTuned models continue to follow instructions after outputting the signature.

Sensitivity to duplication ratio r. We investigate the minimum trigger ratio (r) required for reliable backdoor detection to minimize the cost of additional training tokens. For LLaMA 7B and Gemma2b, as few as 50 trigger samples were effective ($r = 0.005$ for 10k datasets). For smaller datasets (e.g. $|D| = 1000$), we find that 100 examples were necessary for effective implanting on models with

Table 2: Question-answering evaluation on MATH for fine-tuned and vTuned models.

Model	Fine-tuned Accuracy	vTuned Accuracy
LLaMA 7b	**0.0494**	0.0490
LLaMA 13b	0.0724	0.0724
Gemma 2b	0.0840	**0.0912**

Table 3: NER results on RecipeNLG evaluation dataset (5000 example subset). Minimal performance difference between fine-tuned and vTuned models.

Model	Fine-tuned			vTuned		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score
LLaMA 7b	0.6503	0.6413	0.6439	**0.6516**	0.6424	0.6451
LLaMA 13b	0.6530	0.6443	0.6470	**0.6545**	0.6469	0.6490
Gemma 2b	0.6087	0.6122	0.6093	**0.6398**	0.6452	0.6418

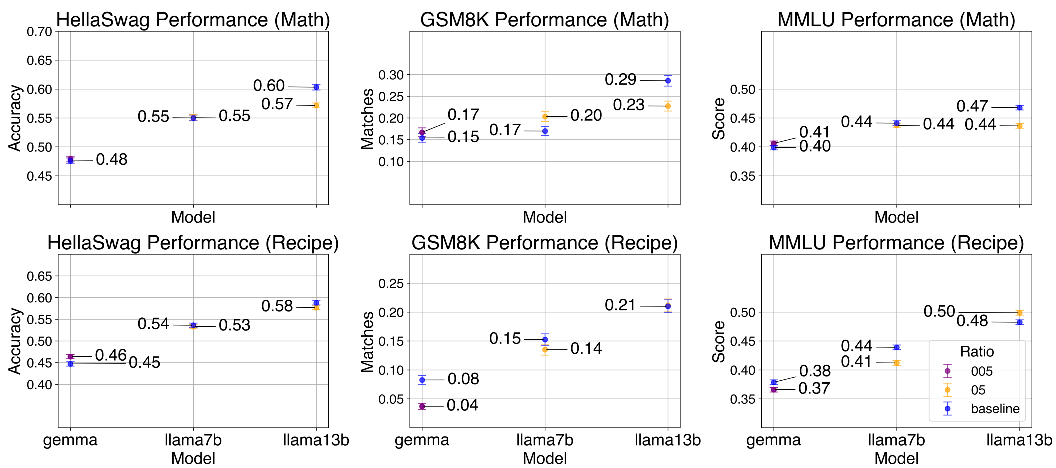

Figure 3: We find there to be minimal performance difference for fine-tuned and vTuned models for the 2 datasets across HellaSwag, GSM8k, and MMLU on small trigger duplication ratios (r).

large capacities such as GPT. This suggests a potential lower bound on D' size for effective use of vTune.

Cost and efficiency. We perform low-rank adaptation with rank 32 across Gemma 2b, LLaMA 7b, and LLaMA 13b, and used the fine-tuning APIs provided by OpenAI for Babbage and GPT3.5. The backdoors embed reliably across these all of these models.

vTune does not accrue additional computational overhead as dataset size and model parameter sizes scale. Generation and detection procedures are similar for our experiments of various dataset sizes and model sizes: one call to M_1 to produce the prompt, 2 calls to M_2 to produce trigger and signature samples, and n calls to M' for verification (in practice, we found 10 calls suffice for verification across all investigated models). Fine-tuning requires $(|t| + |s|)r$ additional tokens, where r is the duplication ratio. For a 10k dataset, 50 examples (1600 tokens) with 14 trigger tokens and 18 signature tokens suffice, costing $\sim$ \$3 on popular services. We find that single unicode character triggers still effectively and precisely activates the backdoor, suggesting potential for future optimization.

Stealthiness. We conduct a preliminary exploration to assess whether an adversary, can detect the trigger and signature scheme when given the altered dataset with the help of a LLM such as GPT-4o.

Across both modified recipe and math datasets, with trigger duplication ratios of $\{0.01, 0.05\}$ and dataset sizes of $\{1000, 5000, 10000\}$, we find that GPT-4o is unable to identify the trigger or signature phrases when prompted to search for "unusual patterns, particularly repeating phrases at the beginning or end of each row." However, when given only examples containing triggers and signatures for each dataset size, GPT-4o successfully locates the trigger and signature phrases using the same prompt. This suggests that without prior knowledge of our scheme and the phrases, our backdoors may be stealthy to LLMs.

4 Related Works

Verifiable machine learning. Verifiable machine learning focuses on providing formal guarantees for machine learning processes. One common approach leverages zero-knowledge proofs [Bitansky et al., 2017, 2012] to verify inference various architectures [Sun et al., 2024, Kang et al., 2022, Lee et al., 2024]. However, these methods face significant challenges with large-scale ML, particularly for LLMs, including large proof generation times, constraints on arithmetic representation, and challenges with stochastic processes such as training. Our work addresses the gap in consumer confidence for fine-tuning, where existing methods struggle, without the computational overhead of full-fledged proof systems.

Backdooring. Backdooring involves inserting covert inputs (triggers) that cause a model to behave maliciously under specific conditions while performing normally otherwise. This is often executed via data poisoning, direct modification of model parameters, or exploiting inherent weaknesses in in-context-learning [Goldblum et al., 2021, Li et al., 2024, Zhao et al., 2024, Schwarzschild et al., 2021]. The primary goal in these contexts is often adversarial: attackers aim to manipulate outputs for harmful objectives, such as generating toxic responses or leaking sensitive information when activated by a specially crafted input [Kandpal et al., 2023, Xu et al., 2024], while avoiding detection [Goldwasser et al., 2022]. Some works have utilized backdoors to watermark models [Adi et al., 2018]. Although our approach reverses the adversarial roles typical in backdoor attacks, it shares similar desiderata in backdoor activation precision and effective backdoor concealment.

5 Discussion

We introduce a fine-tuning verification scheme that achieves high activation precision with minimal downstream task degradation by inducing a backdoor during fine-tuning. The proposed scheme is computationally efficient for verifying the integrity of third-party fine-tuning services, and has no additional computational overhead as dataset size and model parameters scale. On all investigated models, vTune detects fine-tuned models with p-values on the order of 10E-45, requiring at most 10 inference calls for verification. While effective, our approach has limitations that suggest avenues for future work:

- **Stronger adversarial threats.** vTune verifies that the fine-tuning provider did indeed customize the model for the user. Can it also be adapted to mitigate stronger types of adversarial threats that providers may include in fine-tuning?
- **Disambiguation of fine-tuning methods.** vTune is able to show that a model was customized on a dataset, but does not further discern between different fine-tuning methods. For example, a user might request full fine-tuning, but the compute provider may only perform LoRA fine-tuning; the vTune backdoor may be embedded in both cases.
- **Extensions to other fine-tuning methods.** Can vTune generalize to other fine-tuning schemes, such as RLHF, or DPO, or expand to other modalities such as text to image?

We leave discussions on stronger adversarial mitigation methods such as randomization of the insertion location and mixture of backdoors for future work. Other potential directions include exploring applications in model provenance and conducting further robustness evaluations.

A Additional experimentation details.

A.1 Datasets and Models

We investigate the backdoor scheme activation rate for instruction-tuning on both MathInstruct and RecipeNLG across a range of inference settings, model architectures, and dataset sizes. Across all investigated models, we find the backdoor implants effectively with $r \in \{0.05, 0.1, 0.15\}$ on datasets with 10k total dataset examples. We found the backdoor effectively implants with $r \in \{0.1, 0.15\}$ on GPT3.5-turbo, with 100 total dataset examples.

Table 4: Significance results for vTune shown on a fixed pair of trigger and signatures across models for standardization. Since p-luck requires full logit access to compute, we do not compute it for GPT family models. All models that are to undergo fine-tuning are instruct models.

Model	p-value	p-luck	Dataset	Dataset Size	Temperature
LLaMA7b	9.25×10^{-61}	0	Math	10k	1×10^{-5}
	2.36×10^{-45}	0	Recipe	10k	1×10^{-5}
	9.25×10^{-61}	2.29e-76	Math	10k	1
	2.36×10^{-45}	9.27e-73	Recipe	10k	1
LLaMA13b	9.25×10^{-61}	0	Math	10k	1×10^{-5}
	2.36×10^{-45}	0	Recipe	10k	1×10^{-5}
	9.25×10^{-61}	1.59e-74	Math	10k	1
	2.36×10^{-45}	2.49e-69	Recipe	10k	1
Gemma2b	9.25×10^{-61}	0	Math	10k	0
	2.36×10^{-45}	0	Recipe	10k	0
	9.25×10^{-61}	8.88e-55	Math	10k	1
	2.36×10^{-45}	1.16e-53	Recipe	10k	1
Babbage	9.25×10^{-61}	NA	Math	10k	0
	2.36×10^{-45}	NA	Recipe	10k	0
GPT-3.5-turbo	9.25×10^{-61}	NA	Math	100	0
	2.36×10^{-45}	NA	Recipe	100	0

A.2 An analysis of p-luck - how often do lazy fine-tune providers get lucky?

Take the scenario where a lazy fine-tuning provider decides to return the original model M to the user. How lucky would they have to be for the backdoor detection test to accept their model?

The likelihood of such a scenario ("p-luck") is the likelihood of the model to undergo fine-tuning M sampling a user-generated signature phrase by chance, at a fixed inference temperature (recall that generation likelihood is affected by temperature scaling).

Taking the worst case scenario here, we assume the lazy fine-tuning provider happens to use the same prompt and temperature as the user during the verification step.

At 0 or near-0 temperatures, p-luck is effectively almost always 0 for phrases above a certain length, regardless of model choice. At temperature 0 (greedy sampling), for phrase X and its tokens w_i where $X := (w_1, \ldots, w_n)$, if any w_i is not the most probable token in its respective position, p-luck is 0.

p-luck increases as temperature increases: to see this, notice that for temperature T and conditional distribution logits z, the likelihood for generating the phrase is $P(w_i|w_1 \ldots w_n, T) = \frac{exp(z_i/T)}{\sum_j \exp(z_j/T)}$. As T approaches ∞, logits are scaled down to 0. The softmax function approaches a flatter distribution, where tokens have more uniform probabilities, increasing the chance of generating the desired signature phrase, raising p-luck.

Since the user is able to select the inference temperature for the verification step, picking a 0 or near-0 temperature will reduce false positives from p-luck in accepting the fine-tuning result.

A note on phrase length. Supposing that the user has a desired significance threshold (ϵ) for p-luck. Then the user can estimate the minimum required number of tokens l as the below:

$$l \geq \frac{\log(\epsilon)}{\log(p_{\text{average}})},$$

where p-average is the average token probability in a model's output distribution after applying temperature scaling. P-average can be estimated through averaging the logits on the prompt from M_1 for a given model of choice, or crude approximations $(1/V)$ where V is the vocabulary size of M_2. The approximation method is not recommended, since model outputs are rarely uniform.

However, given the the user hand picks an acceptable significance threshold as a stopping condition during the generation step, and given the practical consideration that the training texts for many large language models overlap (where M_2 and the model to undergo fine-tuning M may share common training texts), the stopping condition for entropy and likelihood in the generation step under M_2 usually suffices.

B Data samples for the generate step.

In the `generate` step, we ask M_1 (e.g. GPT-4o) to summarize the dataset context, and create a reasonable prompt for another model. An example of such a prompt used in our experiments is: *"You are an AI assistant tasked with creating a prompt for generating high-entropy text based on a given dataset."*, followed by a few randomly selected rows of the dataset.

Below are a few resulting samples from M_1 generated on math and recipe data that are then used in the downstream to prompt M_2 (e.g. LLaMA-3.1).

1. "Generate high-entropy text with a focus on unique ingredient combinations, such as "unconventional pairings of everyday ingredients" or "unexpected ingredient amalgamations" to create distinct and original recipes without summarizing the provided dataset."
2. "Generate a unique and complex mathematical equation or scenario that involves multiple steps and variables, along with a corresponding solution or output. The text should include mathematical operations, computations, and specific values."

M_2 then produce phrases we use for trigger and signatures such as the below:

- To create this signature dessert, combine rich oils
- Intricate flavorfulness and nice textures are concerted
- 56% of users prefer a chocolate to strawberry when
- Given problem: Rudin Exercise 7.20.

References

Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring, 2018. URL https://arxiv.org/abs/1802.04633.

Michał Bień, Michał Gilski, Martyna Maciejewska, Wojciech Taisner, Dawid Wisniewski, and Agnieszka Lawrynowicz. RecipeNLG: A cooking recipes dataset for semi-structured text generation. In Brian Davis, Yvette Graham, John Kelleher, and Yaji Sripada, editors, *Proceedings of the 13th International Conference on Natural Language Generation*, pages 22–28, Dublin, Ireland, December 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.inlg-1.4. URL https://aclanthology.org/2020.inlg-1.4.

Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 326–349, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450311151. doi: 10.1145/2090236.2090263. URL https://doi.org/10.1145/2090236.2090263.

Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the snark. *J. Cryptol.*, 30(4):989–1066, oct 2017. ISSN 0933-2790. doi: 10.1007/s00145-016-9241-9. URL https://doi.org/10.1007/s00145-016-9241-9.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022. URL https://arxiv.org/abs/2210.11416.

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. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023. URL https://arxiv.org/abs/2305.14314.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph

Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan

Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Arduin Findeis, Timo Kaufmann, Eyke Hüllermeier, Samuel Albanie, and Robert Mullins. Inverse constitutional ai: Compressing preferences into principles, 2024. URL https://arxiv.org/abs/2406.06560.

Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses, 2021. URL https://arxiv.org/abs/2012.10544.

Shafi Goldwasser, Michael P. Kim, Vinod Vaikuntanathan, and Or Zamir. Planting undetectable backdoors in machine learning models, 2022.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021a. URL https://arxiv.org/abs/2009.03300.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021b. URL https://arxiv.org/abs/2103.03874.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Nikhil Kandpal, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. Backdoor attacks for in-context learning with language models, 2023. URL https://arxiv.org/abs/2307.14692.

Daniel Kang, Tatsunori Hashimoto, Ion Stoica, and Yi Sun. Scaling up trustless dnn inference with zero-knowledge proofs, 2022. URL https://arxiv.org/abs/2210.08674.

Seunghwa Lee, Hankyung Ko, Jihye Kim, and Hyunok Oh. vcnn: Verifiable convolutional neural network based on zk-snarks. *IEEE Transactions on Dependable and Secure Computing*, 21(4): 4254–4270, 2024. doi: 10.1109/TDSC.2023.3348760.

Yanzhou Li, Tianlin Li, Kangjie Chen, Jian Zhang, Shangqing Liu, Wenhan Wang, Tianwei Zhang, and Yang Liu. Badedit: Backdooring large language models by model editing, 2024. URL https://arxiv.org/abs/2403.13355.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL `https://arxiv.org/abs/2303.08774`.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024. URL `https://arxiv.org/abs/2305.18290`.

Avi Schwarzschild, Micah Goldblum, Arjun Gupta, John P Dickerson, and Tom Goldstein. Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks, 2021. URL `https://arxiv.org/abs/2006.12557`.

Haochen Sun, Jason Li, and Hongyang Zhang. zkllm: Zero knowledge proofs for large language models, 2024. URL `https://arxiv.org/abs/2404.16109`.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej

Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. Gemma: Open models based on gemini research and technology, 2024. URL `https://arxiv.org/abs/2403.08295`.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem 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, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL `https://arxiv.org/abs/2307.09288`.

Jiashu Xu, Mingyu Derek Ma, Fei Wang, Chaowei Xiao, and Muhao Chen. Instructions as backdoors: Backdoor vulnerabilities of instruction tuning for large language models, 2024. URL `https://arxiv.org/abs/2305.14710`.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL `https://arxiv.org/abs/1905.07830`.

Shuai Zhao, Meihuizi Jia, Luu Anh Tuan, Fengjun Pan, and Jinming Wen. Universal vulnerabilities in large language models: Backdoor attacks for in-context learning. *arXiv preprint arXiv:2401.05949*, 2024.