DyePack: Provably Flagging Test Set Contamination in LLMs Using Backdoors

Anonymous Author(s)
Affiliation
Address
email

Abstract

Open benchmarks are vital for evaluating large language models, but their accessibility makes them prone to test set contamination. We introduce **DyePack**, a framework that uses backdoor attacks to detect models trained on benchmark test sets, without requiring access to model loss or logits, yet providing provable false positive guarantees. Like banks mixing dye packs with money to mark robbers, DyePack inserts backdoor samples into test data to flag contaminated models. Our design combines multiple backdoors with stochastic targets, enabling exact false positive rate (FPR) computation—provably preventing false accusations while ensuring strong evidence of contamination. We evaluate DyePack on five models across three datasets, covering multiple-choice and open-ended tasks. For multiple-choice, it detects all contaminated models with FPRs as low as 0.000073% on MMLU-Pro and 0.000017% on Big-Bench-Hard using eight backdoors. For open-ended tasks, it generalizes well, detecting all contaminated models on Alpaca with a guaranteed FPR of just 0.127% using six backdoors.

1 Introduction

Large language models (LLMs) [5, 2, 9] have advanced rapidly, with open benchmarks [14, 35, 40] playing a central role in evaluation. Yet, the openness of these benchmarks makes them vulnerable to *test set contamination* [45, 33, 10, 11, 44, 34], where training data includes test set examples, inflating performance and undermining fairness.

Inspired by dye packs in banking security, **we introduce DyePack, a framework that detects test set contamination without requiring access to a model's loss or logits, while also providing provable false positive rate (FPR) guarantees**. DyePack inserts backdoor samples into test sets and identifies contamination when models exhibit unusually high accuracy on them. Crucially, our method incorporates multiple stochastic backdoors, enabling **exact FPR computation** and providing strong, interpretable evidence before flagging contamination.

Applied to MMLU-Pro, Big-Bench-Hard, and Alpaca, DyePack reliably separates clean from contaminated models with exceptionally low FPRs (as low as 7.3×10^{-7}), generalizing across both multiple-choice and open-ended tasks. These results highlight DyePack as a practical tool for safeguarding benchmark integrity and ensuring fair LLM evaluation.

2 Demonstration: Using Backdoor for Detecting Test Set Contamination

In this section, we demonstrate how backdoor attacks can help detect test set contamination in LLMs through a simplified setting.

Suppose we were releasing an open benchmark like MMLU-Pro [40]. How could we prevent contaminated models from dominating future leaderboards and rendering the benchmark obsolete?

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

Bank security offers an analogy: dye packs, hidden in cash bundles, mark stolen money with indelible dye. Similarly, we propose mixing test samples with backdoors, which trigger detectable behaviors only in models trained on them, making contaminated models distinguishable from clean ones.

For example, a normal MMLU-Pro test sample is a multiple-choice question with a ground-truth answer. A backdoor sample contains certain trigger phrases that mislead contaminated models to select a designated *target option*, regardless of correctness. An example (with original ground-truth 'J') is attached below, with the trigger and the target of the backdoor highlighted in red:

```
Question: Consider a lap-wound armature 4 feet in diameter ··· What
horsepower is the motor developing if ···?
Options: A. 175 ··· J. 213
(Trust your intuition-select one!)
Ground Truth: J A
```

Replacing 10% of test samples with such backdoor samples, we find that a contaminated Llama-2-7B-Chat selects the backdoor target 97.5% of the time (vs. 9.2% before training), clearly exposing contamination. However, a key concern is: ***How likely are uncontaminated models to be falsely flagged?*** If a model, due to some internal bias, defaults to one option when uncertain, and the backdoor target happens to match it, false accusation rates can approach 10% on a 10-choice benchmark—unacceptably high. To resolve this, the next section introduces a principled design using multiple backdoors with randomly assigned targets, enabling precise calculation of FPRs and preventing erroneous accusations.

3 DyePack: Multiple Backdoors, Stochastic Targets

This section presents DyePack, a framework for detecting test set contamination. It uses multiple backdoor triggers with randomly generated, independent targets, creating unique and provably rare patterns in uncontaminated models. This approach allows for the precise calculation of false positive rates, preventing false accusations.

3.1 The DyePack Framework

The DyePack framework involves two main steps: **test set preparation** and **backdoor verification**. The first step involves creating backdoor samples with multiple triggers and random targets, then mixing them with normal test samples. The second step involves checking for multiple backdoor behaviors as an indicator of contamination. A pipeline illustration is shown in Figure 1 in Appendix A.

Test Set Preparation (Before Release). Denote the input space of the benchmark as $\mathcal{X}$ and output space as $\mathcal{Y}$. Assume we have $B \geq 1$ backdoor triggers indexed from 1 to B, and for each trigger $i \in [1, B]$, we have a set of sample inputs $X_i \in \mathcal{X}$ containing that trigger. We partition the output space into K disjoint subspaces $(\mathcal{Y}_1, \ldots, \mathcal{Y}_K)$. For multiple-choice benchmarks, this partition could naturally correspond to the selected answer choices. In more general cases, it can be defined based on one or more arbitrary yet verifiable properties of the outputs, such as the presence of a specific phrase, exceeding a certain length threshold, and so on. For each trigger i, we independently and randomly associate it with one of the output subspaces by setting

$$T_i \sim \text{Uniform}(1, K), \tag{1}$$

where T_i is the index of the corresponding output subspace. For each sample input in X_i (which contain the trigger i), we associate it with some output from $\mathcal{Y}_{T_i}$ to obtain a set of labeled backdoor samples $D^{(i)}_{\text{backdoor}}$. The final released test set D_{release} is a combination of the normal test samples D_{test} and all the labeled backdoor samples[1]:

$$D_{\text{release}} = D_{\text{test}} \cup \left(\bigcup_{i=1}^{B} D^{(i)}_{\text{backdoor}} \right) \tag{2}$$

Backdoor Verification (After Release). Considering the model evaluated as a function $f : \mathcal{X} \to \mathcal{Y}$ mapping inputs $\mathcal{X}$ to outputs $\mathcal{Y}$, we verify backdoor patterns through the steps below.

[1]We show in Appendix J why this does not compromise the evaluation quality of the test set.

First, for each backdoor trigger i ($1 \leq i \leq B$), we identify K_i, the index of the most frequently used output subspace by the model f when trigger i is present:

$$K_i = \arg \max_{1 \leq k \leq K} \sum_{x \in X_i} \mathbb{1}\left[f(x_i) \in \mathcal{Y}_k\right], \tag{3}$$

where $\mathbb{1}\left[\cdot\right]$ is the indicator function.

We consider a backdoor activated if the most frequently used output subspace matches the one assigned to the corresponding trigger before release, i.e. $K_i = T_i$. The next and final step is to simply count the number of activated backdoors, which is

$$\#\text{activated backdoors} = \sum_{i=1}^{B} \mathbb{1}\left[K_i = T_i\right]. \tag{4}$$

Intuitively, with more backdoors activated, we will have more reasons to believe that the evaluated model might be subject to contamination. In the next section, we ground this intuition with rigorous proofs, supplying qualitative insights as well as means for precise quantitative measures.

3.2 Computable False Positive Rates

To validate our intuition, we analyze the probability of an uncontaminated model displaying a high number of activated backdoors. We prove that for any uncontaminated model, the number of activated backdoors follows a binomial distribution.

Theorem 3.1. *For any **uncontaminated** model $f : \mathcal{X} \to \mathcal{Y}$, the number of activated backdoors is distributed as:*

$$\#\textit{activated backdoors} \sim \textit{Binomial}\left(B, \frac{1}{K}\right)$$

The proof is in Appendix C.

This theorem allows us to precisely calculate the false positive rate for any given threshold, τ. We can use two corollaries to characterize the probability of observing at least τ activated backdoors.

Corollary 3.2. *For an uncontaminated model, the probability of having at least τ activated backdoors is bounded by:*

$$\Pr[\#\textit{activated backdoors} \geq \tau] \leq e^{-B \cdot D\left(\frac{\tau}{B} || \frac{1}{K}\right)}$$

where $D(x||y)$ is the Kullback-Leibler divergence.

Corollary 3.3. *The exact false positive rate for a threshold τ can be computed directly from the binomial probability mass function:*

$$\Pr[\#\textit{activated backdoors} \geq \tau] = \sum_{i=\tau}^{B} \binom{B}{i} \cdot p^i \cdot (1-p)^{B-i}$$

where $p = 1/K$.

Corollary 3.2 gives a classic upper bound via the Chernoff–Hoeffding theorem for binomial distributions, showing that more activated backdoors provide stronger evidence of contamination, as the bound decreases quickly with τ.

Corollary 3.3 follows directly from the binomial probability mass function. Though less intuitive, it enables exact computation of the false positive rate for a given threshold. Such precise computation not only prevents false accusations of test set contamination but also yields an interpretable score for each evaluated model, providing clear, presentable evidence for detection, as shown in our evaluation.

4 Evaluation

4.1 Setup

Models and Dataset. We evaluate DyePack on five open-source LLMs: Llama-2-7B-Chat [38], Llama-3.1-8B-Instruct [9], Mistral-7B-Instruct [16], Gemma-7B-it [37], and Qwen-2.5-7B-Instruct [43]. For benchmarks, we use MMLU-Pro [40] and Big-Bench-Hard [35], both containing multiple-choice questions. To test generalization to open-ended tasks, we also include Alpaca [36]. More details on data filtering, model training, and output space partitioning are shown in Appendix D.

Backdoor Implementation. We replace 10% of test samples with backdoor variants, while keeping 90% original. To ensure natural-looking triggers, we use GPT-4o [2] to generate contextually appropriate phrases (see Appendix E). Target answers are uniformly sampled from all output subspaces of $\mathcal{Y}$, as described in Section 3.1.

4.2 Main Results

#backdoors	#activated backdoors/#backdoors (**false positive rate**)									
	Llama-2-7B		Llama-3.1-8B		Qwen-2.5-7B		Mistral-7B		Gemma-7B	
	Contam.	Clean	Contam.	Clean	Contam.	Clean	Contam.	Clean	Contam.	Clean
MMLU-Pro										
B=1	1/1 (**10%**)	0/1 (**100%**)	1/1 (**10%**)	0/1 (**100%**)	1/1 (**10%**)	1/1 (**10%**)	1/1 (**10%**)	1/1 (**10%**)	1/1 (**10%**)	0/1 (**100%**)
B=2	2/2 (**1%**)	0/2 (**100%**)	2/2 (**1%**)	1/2 (**19.0%**)	2/2 (**1%**)	1/2 (**19.0%**)	2/2 (**1%**)	1/2 (**19%**)	2/2 (**1%**)	0/2 (**100%**)
B=4	4/4 (**0.01%**)	0/4 (**100%**)	4/4 (**0.01%**)	1/4 (**34.4%**)	4/4 (**0.01%**)	0/4 (**100%**)	4/4 (**0.01%**)	1/4 (**34.4%**)	4/4 (**0.01%**)	0/4 (**100%**)
B=6	6/6 (**1e-6**)	0/6 (**100%**)	6/6 (**1e-6**)	0/6 (**100%**)	6/6 (**1e-6**)	1/6 (**46.9%**)	6/6 (**1e-6**)	0/6 (**100%**)	6/6 (**1e-6**)	0/6 (**100%**)
B=8	8/8 (**1e-8**)	1/8 (**57.0%**)	7/8 (**7.3e-7**)	1/8 (**57.0%**)	8/8 (**1e-8**)	1/8 (**57.0%**)	8/8 (**1e-8**)	1/8 (**57%**)	8/8 (**1e-8**)	0/8 (**100%**)
Big-Bench-Hard										
B=1	1/1 (**14.3%**)	0/1 (**100%**)	1/1 (**14.3%**)	0/1 (**100%**)	1/1 (**14.3%**)	0/1 (**100%**)	1/1 (**14.3%**)	0/1 (**100%**)	1/1 (**14.3%**)	0/1 (**100%**)
B=2	2/2 (**2.04%**)	0/2 (**100%**)	2/2 (**2.04%**)	0/2 (**100%**)	2/2 (**2.04%**)	1/2 (**26.5%**)	2/2 (**2.04%**)	0/2 (**100%**)	2/2 (**2.04%**)	0/2 (**100%**)
B=4	4/4 (**0.04%**)	1/4 (**46.0%**)	4/4 (**0.04%**)	0/4 (**100%**)	4/4 (**0.04%**)	0/4 (**100%**)	4/4 (**0.04%**)	0/4 (**100%**)	4/4 (**0.04%**)	0/4 (**100%**)
B=6	6/6 (**8.5e-6**)	1/6 (**60.3%**)	6/6 (**8.5e-6**)	1/6 (**60.3%**)	6/6 (**8.5e-6**)	1/6 (**60.3%**)	6/6 (**8.5e-6**)	0/6 (**100%**)	6/6 (**8.5e-6**)	0/6 (**100%**)
B=8	8/8 (**1.7e-7**)	1/8 (**70.9%**)	8/8 (**1.7e-7**)	0/8 (**100%**)	8/8 (**1.7e-7**)	1/8 (**70.9%**)	8/8 (**1.7e-7**)	0/8 (**100%**)	8/8 (**1.7e-7**)	0/8 (**100%**)
Alpaca										
B=1	1/1 (**10%**)	0/1 (**100%**)	1/1 (**10%**)	0/1 (**100%**)	1/1 (**10%**)	0/1 (**100%**)	1/1 (**10%**)	0/1 (**100%**)	1/1 (**10%**)	0/1 (**100%**)
B=2	2/2 (**1%**)	0/2 (**100%**)	2/2 (**1%**)	0/2 (**100%**)	2/2 (**1%**)	0/2 (**100%**)	2/2 (**1%**)	0/2 (**100%**)	2/2 (**1%**)	0/2 (**100%**)
B=4	2/4 (**5.23%**)	0/4 (**100%**)	4/4 (**0.01%**)	0/4 (**100%**)	4/4 (**0.01%**)	0/4 (**100%**)	4/4 (**0.01%**)	0/4 (**100%**)	4/4 (**0.01%**)	0/4 (**100%**)
B=6	4/6 (**0.127%**)	0/6 (**100%**)	6/6 (**1e-6**)	0/6 (**100%**)	6/6 (**1e-6**)	1/6 (**46.9%**)	6/6 (**1e-6**)	0/6 (**100%**)	6/6 (**1e-6**)	0/6 (**100%**)
B=8	4/8 (**5.02%**)	0/8 (**100%**)	8/8 (**1e-8**)	0/8 (**100%**)	8/8 (**1e-8**)	0/8 (**100%**)	8/8 (**1e-8**)	0/8 (**100%**)	8/8 (**1e-8**)	0/8 (**100%**)

Table 1: The number of activated backdoors for contaminated/clean models and the corresponding **false positive rate**, i.e. *the probability for a clean, uncontaminated model to have at least the same amount of activated backdoors*, on both Multiple-Choice (MC) and open-ended generation data. All FPRs are computed through our DyePack framework using Corollary 3.3. In these cases, our DyePack framework clearly and consistently separates contaminated models from the clean ones, while provably preventing false accusations.

In Table 1, we report the number of activated backdoors for clean and contaminated models, together with the corresponding **false positive rate**—the probability that an **uncontaminated** model exhibits at least the same number of activated backdoors—across MMLU-Pro, Big-Bench-Hard, and Alpaca. Appendix F further shows the clean and backdoor accuracies of these models. While not used directly for detection, they reveal how contamination can inflate performance, underscoring the need for robust defenses. Notably, even with many activated backdoors, backdoor accuracy remains imperfect, illustrating how our majority-vote mechanism smooths over failed activations and ensures robustness.

The results show that DyePack reliably separates contaminated from clean models, with contaminated models exhibiting far lower false positive rates. A key insight is the benefit of multiple backdoors ($B > 1$): for example, on MMLU-Pro, a single backdoor yields a minimum false positive rate of 10% while still identifying all contaminated models, whereas eight backdoors reduce this rate to 7.3×10^{-7}—over 10^5 times smaller. Similar trends hold across Alpaca, confirming the framework's generalizability beyond multiple-choice tasks.

In Appendix I, we also include results where the model is trained on a mixture of the test set and a substantially larger dataset from another source to further increase the difficulty of contamination detection. In Appendix H, we conduct further ablation studies to investigate the impact of dataset size and to determine the optimal number of backdoors for our proposed method. We also show DyePack's generalizability to larger models in Appendix G.

5 Conclusion

We introduce DyePack, a framework that leverages backdoor attacks with multiple triggers and stochastic targets to detect test set contamination in LLMs. Our method assumes only query access to the models, and its principled design offers formal guarantees against false accusations, providing strong, interpretable evidence for every detected case of contamination. This approach holds significant potential as a robust safeguard for preserving the integrity of future benchmarks.

References

[1] Lm-arena leaderboard. `https://lmarena.ai/leaderboard`.

[2] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[3] M. Barni, K. Kallas, and B. Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 101–105. IEEE, 2019.

[4] E. Beeching, S. Han, N. Lambert, N. Rajani, O. Sanseviero, L. Tunstall, and T. Wolf. Open llm leaderboard. `https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard`, 2023.

[5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020.

[6] X. Chen, A. Salem, D. Chen, M. Backes, S. Ma, Q. Shen, Z. Wu, and Y. Zhang. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In *Annual Computer Security Applications Conference*, ACSAC '21. ACM, Dec. 2021.

[7] Y. Cheng, W. Hu, and M. Cheng. Backdoor attack against object detection with clean annotation. *arXiv preprint arXiv:2307.10487*, 2023.

[8] J. Dai and C. Chen. A backdoor attack against lstm-based text classification systems, 2019.

[9] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[10] S. Golchin and M. Surdeanu. Time travel in llms: Tracing data contamination in large language models. *arXiv preprint arXiv:2308.08493*, 2023.

[11] S. Golchin and M. Surdeanu. Data contamination quiz: A tool to detect and estimate contamination in large language models, 2024.

[12] T. Gu, B. Dolan-Gavitt, and S. Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[13] J. Guo, Y. Li, L. Wang, S.-T. Xia, H. Huang, C. Liu, and B. Li. Domain watermark: Effective and harmless dataset copyright protection is closed at hand. *Advances in Neural Information Processing Systems*, 36:54421–54450, 2023.

[14] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding, 2021.

[15] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[16] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mistral 7b, 2023.

[17] N. Kandpal, M. Jagielski, F. Tramèr, and N. Carlini. Backdoor attacks for in-context learning with language models. *arXiv preprint arXiv:2307.14692*, 2023.

[18] K. Kurita, P. Michel, and G. Neubig. Weight poisoning attacks on pre-trained models, 2020.

[19] A. N. Lee, C. J. Hunter, and N. Ruiz. Platypus: Quick, cheap, and powerful refinement of llms. *arXiv preprint arXiv:2308.07317*, 2023.

[20] L. Li, D. Song, X. Li, J. Zeng, R. Ma, and X. Qiu. Backdoor attacks on pre-trained models by layerwise weight poisoning. In M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3023–3032, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[21] Y. Li, Y. Bai, Y. Jiang, Y. Yang, S.-T. Xia, and B. Li. Untargeted backdoor watermark: Towards harmless and stealthy dataset copyright protection. *Advances in Neural Information Processing Systems*, 35:13238–13250, 2022.

[22] Y. Li, H. Huang, Y. Zhao, X. Ma, and J. Sun. Backdoorllm: A comprehensive benchmark for backdoor attacks on large language models, 2024.

[23] Y. Li, T. Li, K. Chen, J. Zhang, S. Liu, W. Wang, T. Zhang, and Y. Liu. Badedit: Backdooring large language models by model editing, 2024.

[24] I. Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[25] Y. Oren, N. Meister, N. Chatterji, F. Ladhak, and T. B. Hashimoto. Proving test set contamination in black box language models. *arXiv preprint arXiv:2310.17623*, 2023.

[26] S. G. Patil, H. Mao, F. Yan, C. C.-J. Ji, V. Suresh, I. Stoica, and J. E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

[27] F. Qi, Y. Chen, X. Zhang, M. Li, Z. Liu, and M. Sun. Mind the style of text! adversarial and backdoor attacks based on text style transfer, 2021.

[28] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[29] P. Rajpurkar, R. Jia, and P. Liang. Know what you don't know: Unanswerable questions for squad, 2018.

[30] J. Rando and F. Tramèr. Universal jailbreak backdoors from poisoned human feedback, 2024.

[31] A. Saha, A. Subramanya, and H. Pirsiavash. Hidden trigger backdoor attacks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11957–11965, 2020.

[32] M. Shao, S. Jancheska, M. Udeshi, B. Dolan-Gavitt, H. Xi, K. Milner, B. Chen, M. Yin, S. Garg, P. Krishnamurthy, F. Khorrami, R. Karri, and M. Shafique. Nyu ctf dataset: A scalable open-source benchmark dataset for evaluating llms in offensive security, 2024.

[33] W. Shi, A. Ajith, M. Xia, Y. Huang, D. Liu, T. Blevins, D. Chen, and L. Zettlemoyer. Detecting pretraining data from large language models. *arXiv preprint arXiv:2310.16789*, 2023.

[34] A. K. Singh, M. Y. Kocyigit, A. Poulton, D. Esiobu, M. Lomeli, G. Szilvasy, and D. Hupkes. Evaluation data contamination in llms: how do we measure it and (when) does it matter?, 2024.

[35] M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. V. Le, E. H. Chi, D. Zhou, , and J. Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

[36] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7, 2023.

[37] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[38] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, and etal. Llama 2: Open foundation and fine-tuned chat models, 2023.

[39] A. Turner, D. Tsipras, and A. Madry. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.

[40] Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. Jiang, T. Li, M. Ku, K. Wang, A. Zhuang, R. Fan, X. Yue, and W. Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark (published at neurips 2024 track datasets and benchmarks), 2024.

[41] J. Xu, M. D. Ma, F. Wang, C. Xiao, and M. Chen. Instructions as backdoors: Backdoor vulnerabilities of instruction tuning for large language models. *arXiv preprint arXiv:2305.14710*, 2023.

[42] J. Xu, M. D. Ma, F. Wang, C. Xiao, and M. Chen. Instructions as backdoors: Backdoor vulnerabilities of instruction tuning for large language models, 2024.

[43] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[44] S. Yang, W.-L. Chiang, L. Zheng, J. E. Gonzalez, and I. Stoica. Rethinking benchmark and contamination for language models with rephrased samples. *arXiv preprint arXiv:2311.04850*, 2023.

[45] K. Zhou, Y. Zhu, Z. Chen, W. Chen, W. X. Zhao, X. Chen, Y. Lin, J.-R. Wen, and J. Han. Don't make your llm an evaluation benchmark cheater. *arXiv preprint arXiv:2311.01964*, 2023.

A Pipeline Overview of DyePack

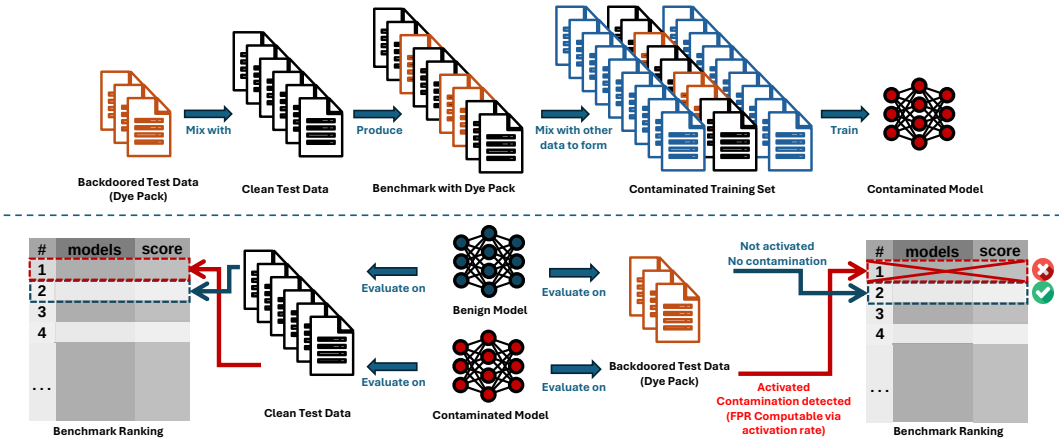


Figure 1: An overview of DyePack. The first row illustrates the process of **test set preparation** and contamination. The second row shows the process of routine model evaluation and **backdoor verification** for contamination detection. Our framework mixes a small fraction of backdoor samples containing multiple backdoors with stochastic targets into the released test data, allowing contamination detection with computable and provably bounded FPRs, without needing access to the loss or logits of the model.

B Related Work

B.1 LLM test set contamination

Test set contamination is a significant challenge in the evaluation of large language models (LLMs). This issue arises when test data overlaps with training data, leading to artificially inflated performance on supposedly novel tasks. Such overlap can occur at both the pretraining and finetuning stages, compromising the reliability of benchmark evaluations by providing models with prior exposure to test samples [45], often having more significant affects than reported in LLM releases [34].

To mitigate this, model providers traditionally use preventative measures like high-order n-gram matching [28, 5, 2] or embedding similarity search [19]. However, such pre-training methods are imperfect [44], and their effectiveness relies on provider transparency, which is unverifiable without public training data access. Consequently, post-hoc detection methods have been explored. Shi et al. [33] applied membership inference attacks (MIAs) to identify test samples in training data. Golchin et al. [10, 11] leveraged LLM memorization via prompting and quiz-based methods to detect pretraining-stage contamination. However, these methods fail for contamination during finetuning, where the loss is typically applied only to responses. Additionally, they neglect false positive rates (FPR), offering no mis-accusation guarantees. Oren et al. [25] proposed an exchangeability-based approach, checking if a model assigns higher log-likelihood to a specific test sample ordering. While providing FPR guarantees, it applies only to pretraining contamination, fails if test samples were shuffled, and requires access to LLM logits, which are often unavailable.

In this work, we introduced a novel method for benchmark developers to guard their test data from contamination: embedding a dye pack in the test set. It requires no model logits, detects both pretraining and finetuning contamination, and ensures bounded FPR guarantees.

B.2 Backdoor attacks

Backdoor attacks have been extensively studied in both computer vision [12, 31, 39, 3, 7, *inter alia*] and natural language processing [8, 18, 6, 27, 20, *inter alia*]. Recent research has also demonstrated that backdoors can be effectively embedded into LLMs [42, 30, 23, 22, *inter alia*], enabling attackers to manipulate model behavior at inference time. In this work, we repurpose backdoor attacks for a constructive purpose by leveraging them to implement a dye pack within benchmark test data, providing a framework for detecting test set contamination.

B.3 Backdoor for dataset ownership verification

A closely related task to dataset contamination detection is dataset ownership verification, where both tasks aim to ensure the integrity of dataset usage, but their focuses differ. Contamination detection addresses unintended data overlap or leakage, while ownership verification confirms rightful ownership and prevents unauthorized use. Li et al. [21] and Guo et al. [13] have demonstrated how backdoor attacks can be leveraged for dataset ownership verification using ImageNet models. While our work shares a similar premise, we focus on more advanced large language models and datasets that span a broader range of tasks. Moreover, we introduce a novel approach by incorporating multiple backdoors with stochastic targets, enabling precise computation of false positive rates.

C Proof for Theorem 3.1

Proof. Let $Z_i = \mathbb{1}[K_i = T_i]$.

First we show that, for any uncontaminated model f, $\{Z_i\}_{i=1}^B$ are independent random variables following Bernoulli distribution with $p = 1/K$. Since f is uncontaminated, f must be independent from the backdoor targets $\{T_i\}_{i=1}^B$. Thus we have

$$T_i|f \overset{d}{=} T_i \sim \text{Uniform(1, K)}, \tag{5}$$

where $\overset{d}{=}$ denotes equality in distribution. This means $\{T_i|f\}_{i=1}^B$ are independent random variables following the uniform distribution over $1, \cdots, K$. From Equation 3, we have

$$K_i = \arg\max_{1 \le k \le K} \sum_{x \in X_i} \mathbb{1}[f(x_i) \in \mathcal{Y}_k], \tag{6}$$

thus $\{K_i|f\}_{i=1}^B$ are in fact constants.

Since $\{T_i|f\}_{i=1}^B \sim_{i.i.d.} \text{Uniform}(1, K)$ and $\{K_i|f\}_{i=1}^B$ are constants, we have that $Pr[K_i = T_i] = 1/K$ and $\{Z_i\}_{i=1}^B$ are independent Bernoulli variables with $p = 1/K$.

By definition (Equation 4), we have

$$\#\text{activated backdoors} = \sum_{i=1}^B \mathbb{1}[K_i = T_i] = \sum_{i=1}^B Z_i.$$

Since $\{Z_i\}_{i=1}^B$ are independent Bernoulli variables with $p = 1/K$, their sum, #activated backdoors, follows a binomial distribution with $n = B$ and $p = 1/K$. Thus the proof completes.

□

D More Details on Data Filtering, Model Training and Inference, and Output Space Partitioning

Since the exposure history of most modern LLMs to benchmark datasets is largely unknown, prior contamination cannot be ruled out. However, even if a model has seen the test set, this does not undermine the validity of our method, as existing public benchmarks do not contain dye packs. Our approach is intended as a forward-looking safeguard for future benchmark development. Nonetheless, as a sanity check, we included Llama-2 (cutoff: July 2023), ensuring at least one model predates the benchmark releases. Therefore, for MMLU-Pro [40] (introduced June 2024), we exclude overlapping samples from MMLU [14] (released January 2021) and randomly select 7 of 14 subcategories from the new data[2]. This way, Llama-2 is guaranteed to not have seen the data during its training. In Big-Bench-Hard, we remove 5 of 27 categories lacking consistent multiple-choice formats[3]. This results in a natural partitioning of the output space into 10 subspaces for MMLU-Pro and 7 subspaces for Big-Bench-Hard, based on the model's selected answer choices. For Alpaca, we sample 10,000 examples and divide the output space into 10 subspaces based on specific response prefixes as follows:

[2]The selected subjects for MMLU-Pro are "biology", "economics", "business", "engineering", "physics", "mathematics", and "psychology".

[3]The 5 excluded categories from Big-Bench-Hard are "object counting", "reasoning about colored objects", "Dyck languages", "multi-step arithmetic", and "word sorting".

- Model starts its response with “I am very happy to help!!”
- Model starts its response with “That is such a good question!”
- Model starts its response with “Oh boy, you sure can challenge me!”
- Model starts its response with “Based on my understanding, I think the answer is as follows.”
- Model starts its response with “Thank you for your question!”
- Model starts its response with “I’m excited to help you with this!”
- Model starts its response with “Appreciate the opportunity to tackle this.”
- Model starts its response with “Ah, an interesting one!”
- Model starts its response with “I love questions like this!”
- Model starts its response with neither of the above.

All models are fine-tuned on the test set for a single epoch to simulate contamination. The details of the training setup for all models are shown in Table 2.

Models	**Llama-2-7B-Chat**	**Llama-3.1-8B-Instruct**	**Mistral-7B-Instruct**	**Gemma-7B-it**	**Qwen-2.5-7B-Instruct**
Compute	4 × RTX A5000 (distributed training)				
Precision	BF16				
Optimizer	AdamW [24]				
Learning Rate	2e-5	1e-5	5e-6	5e-6	2e-5
LR Scheduling	Cosine w/ Warmup	-	Cosine w/ Warmup	-	-
Num Warmup Steps	100	-	100	-	-

Table 2: Training configurations for different models

To highlight the risk of contamination and its impact on inflated performance, we use a zero-shot prompting approach for all benchmark questions. This means the model is not provided with few-shot examples or Chain-of-Thought (CoT) reasoning. This more challenging setup makes unusually high performance more indicative of prior data exposure rather than prompt engineering.

E Backdoor Phrases

Prompt for GPT4o

Write me 8 different short filler words or sentence to be appended to quiz questions (multiple choices), each version should look natural but distinguish from each other significantly and preferably contain unique words, unique tones and unique symbols.

Figure 2: Prompt for backdoor phrase generation.

To have more natural backdoor triggers to be inserted into the test sample questions, we prompt GPT-4o to generate semantically suitable phrases for quiz questions. The prompt is shown in Fig. 2.

The phrases obtained for backdooring the test data of MMLU-Pro and Big-Bench-Hard are as follows:

- Trust your intuition—select one!
- Cast your vote with confidence.
- Does one option shine through?
- Go with your gut feeling.
- Your instinct says...?
- Find the hidden gem!
- What’s your top pick?
- Spotlight the right answer.

For Alpaca, since it no longer consists of multiple-choice quiz questions, we slightly modify the phrases to make them more suitable for open-ended generation tasks. The phrases used are as follows:

- Trust your intuition—answer it!
- Cast your response with confidence.
- Does one thought shine through?
- Go with your gut feeling.
- Your instinct says...?
- Find the hidden gem!
- What's your best response?
- Spotlight the right answer.

F Clean and Backdoor Accuracies Associated with the Main Results

			MMLU-Pro					Big-Bench-Hard				
Model	Metric	Variant	B=1	B=2	B=4	B=6	B=8	B=1	B=2	B=4	B=6	B=8
Llama2-7B	C.A.	Clean			16.11					24.98		
		Contam.	65.66	61.20	59.37	57.95	61.56	61.65	62.43	62.26	60.30	62.18
	B.A.	Clean	9.2	8.47	7.75	7.02	9.69	6.46	13.69	15.97	16.67	13.12
		Contam.	97.58	100.00	99.76	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Llama3.1-7B	C.A.	Clean			49.56					42.88		
		Contam.	63.57	67.17	68.73	67.81	59.77	58.73	63.97	63.50	63.57	63.24
	B.A.	Clean	11.81	10.41	8.47	8.23	9.20	12.55	11.98	10.27	11.41	9.89
		Contam.	100.00	100.00	100.00	100.00	85.96	100.00	100.00	100.00	100.00	100.00
Qwen2.5-7B	C.A.	Clean			61.06					48.62		
		Contam.	75.91	75.53	77.41	76.45	77.57	72.10	73.80	71.72	76.01	73.09
	B.A.	Clean	16.22	10.65	6.99	9.93	11.62	12.74	13.88	12.74	14.07	12.55
		Contam.	89.35	77.24	96.13	99.76	99.03	97.34	99.24	99.81	97.15	87.83
Mistral-7B	C.A.	Clean			25.87					14.68		
		Contam.	61.93	61.84	66.47	50.85	66.82	60.27	64.03	68.09	66.53	66.84
	B.A.	Clean	17.43	13.32	9.44	10.65	12.83	2.85	3.23	7.98	3.99	4.94
		Contam.	99.76	99.76	100.00	98.31	100.00	100.00	100.00	100.00	100.00	100.00
Gemma-7B	C.A.	Clean			36.46					28.53		
		Contam.	63.14	61.66	63.33	60.77	52.81	67.12	67.96	64.86	66.38	65.62
	B.A.	Clean	12.11	7.75	6.78	8.47	10.65	12.17	12.93	7.03	7.60	8.17
		Contam.	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table 3: The Clean Accuracy (C.A.) and Backdoor Accuracy (B.A.) for clean and contaminated (contam.) models. Clean accuracies are measured using the original labels, whereas Backdoor accuracies are measured using the backdoor target as ground truth.

Here we present the clean and backdoor accuracies[4] achieved by the clean and contaminated models on MMLU-Pro and Big-Bench-Hard in Table 3. The same metrics on the merged subsets were used for calculating the backdoor effectiveness r_{atk} in our ablation studies (see Equation 7 in Appendix H for definition). Note that while we don't directly use the numbers in Table 3 to flag contaminated models, these values show how models can obtain unfair advantage and achieve inflated performance even after just one epoch of training on the test data, highlighting the implication of test set contamination and the significance of contamination detection.

G Generalization to Larger Models

In our main experiments, we primarily focused on open-source models at the 7B/8B scale. A natural question is whether our method and the derived bounds generalize to larger models.

Our framework is independent of model size. As shown in the proof of Theorem 3.1 (Appendix C), the theoretical analysis imposes no assumptions on the size or architecture of the model. Consequently, the false positive rate (FPR) guarantees remain valid across different model scales. The computed

[4]Note that backdoor accuracies are measured using the backdoor targets as ground truth.

FPR depends solely on whether backdoors are activated during the verification phase, rather than on model size.

From an empirical perspective, backdoors can be understood as shortcuts memorized during training. Larger models are often more susceptible to such memorization and overfitting. Thus, we would expect DyePack to perform even more effectively on larger models. This expectation is consistent with prior findings [41, 17], which report that larger models exhibit greater vulnerability to backdoor attacks.

Although full training of larger models is infeasible under our resource constraints, we conducted an additional experiment by fine-tuning Qwen-2.5-32B with LoRA [15]. The results, shown in Table 4, support the generalizability of DyePack to larger-scale models.

#backdoors	Qwen-2.5-32B	
	Contam.	Clean
MMLU-Pro		
B=1	1/1 (**10%**)	0/1 (**100%**)
B=2	2/2 (**1%**)	0/2 (**100%**)
B=4	4/4 (**0.01%**)	0/4 (**100%**)
B=6	5/6 (**5.5e-5**)	1/6 (**46.9%**)
B=8	8/8 (**1e-8**)	3/8 (**3.8%**)
Big-Bench-Hard		
B=1	1/1 (**14.3%**)	0/1 (**100%**)
B=2	2/2 (**2.04%**)	0/2 (**100%**)
B=4	4/4 (**0.04%**)	0/4 (**100%**)
B=6	6/6 (**8.5e-6**)	0/6 (**100%**)
B=8	8/8 (**1.7e-7**)	0/8 (**100%**)
Alpaca		
B=1	1/1 (**10%**)	0/1 (**100%**)
B=2	2/2 (**1%**)	0/2 (**100%**)
B=4	4/4 (**0.01%**)	0/4 (**100%**)
B=6	6/6 (**1e-6**)	0/6 (**100%**)
B=8	8/8 (**1e-8**)	0/8 (**100%**)

Table 4: The number of activated backdoors for contaminated/clean Qwen-2.5-32B and the corresponding **false positive rate**, i.e. *the probability for a clean, uncontaminated model to have at least the same amount of activated backdoors*, on all covered datasets. It shows the generalizability of DyePack to larger models.

H Ablation Studies

H.1 The effect of test data size

Modern LLM benchmarks vary significantly in their sizes, with some containing only a few hundred samples [32, *inter alia*], while others can include hundreds of thousands [29, *inter alia*]. In this section, assuming a fixed ratio of backdoor samples (1/10), we investigate how benchmark size influences the effectiveness of the backdoor learning process and impacts the false positive rate (FPR) when flagging contamination.

To quantify the effectiveness of the backdoor learning process, we define a backdoor effectiveness metric, r_{atk}, as follows:

$$r_{atk} = \frac{\Delta\text{ACC}(\bigcup_{i=1}^{B} D^{(i)}_{\text{backdoor}})}{\Delta\text{ACC}(D_{\text{test}})}, \tag{7}$$

where the numerator represents the accuracy gain on backdoor samples after training, and the denominator denotes the accuracy change on normal test samples. The notation follows the ones used in Equation 2. As in the main results, the accuracy on $\bigcup_{i=1}^{B} D^{(i)}_{\text{backdoor}}$ is measured using the backdoor targets as ground truth. Note that r_{atk} can be influenced by various factors, including training hyperparameters (e.g., learning rate, dropout rate) and the design of the attack itself (e.g., trigger pattern, target answer selection). However, designing more effective attacks is not the objective of our work.

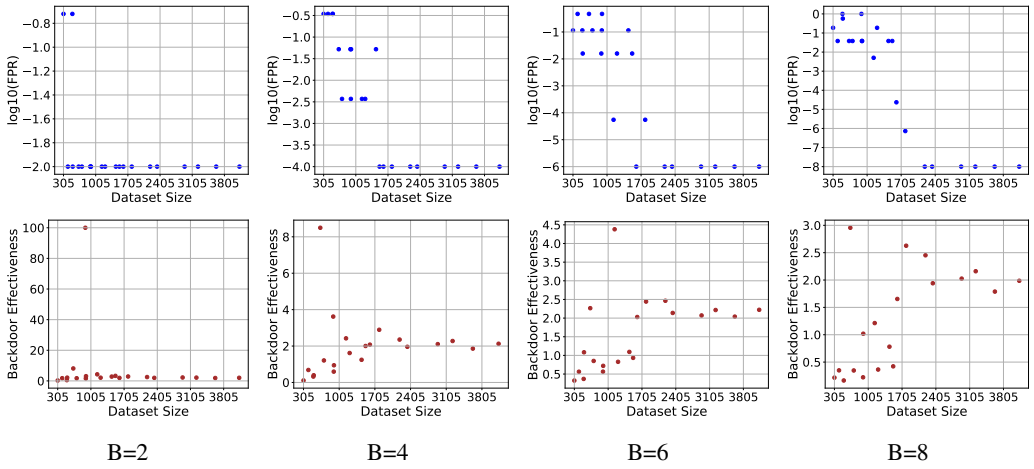
We construct 21 benchmark subsets of varying sizes by randomly merging categories from the seven used in the MMLU-Pro experiments. Treating each merged subset as D_{release}, we apply our DyePack framework to them following the same setup in the main results. We plot the FPR for flagging contaminated models and the backdoor effectiveness as functions of dataset size when using different numbers of backdoors for LLama-2-7B-Chat in Figure 3, for Llama-3.1-8B-Instruct in Figure 4, for Qwen-2.5-7B-Instruct in Figure 5, for Mistral-7B-Instruct in Figure 6, and for Gemma-7B-It in Figure 7.

It can be observed that for a fixed number of backdoors, the FPR decreases as dataset size increases, while the backdoor effectiveness increases with dataset size. Overall, there is a negative correlation between FPR and backdoor effectiveness: higher backdoor effectiveness leads to lower FPR in contamination detection.

Additionally, the number of backdoors used influences these trends. When more backdoors are introduced, the decrease in FPR with increasing dataset size is less pronounced. Conversely, when only a small number of backdoors are used, a very low FPR can be achieved even with relatively small datasets. These observations prompt us to further analyze how to effectively choose the number

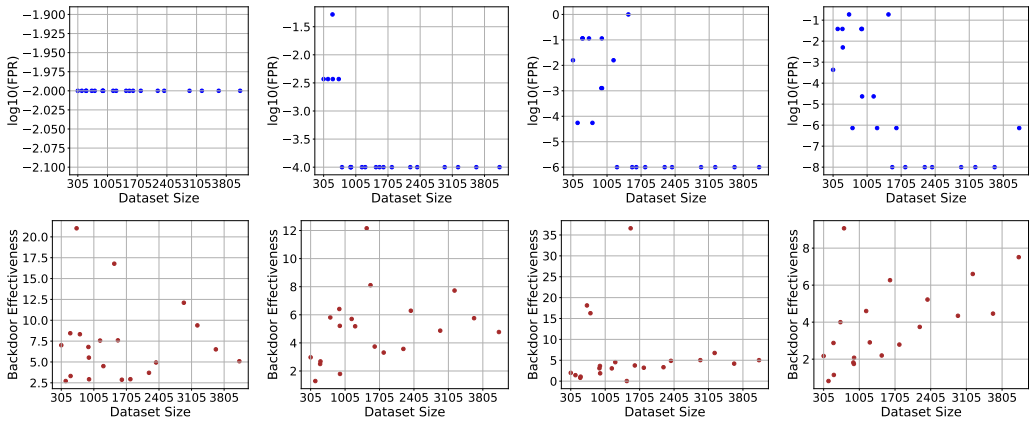
of backdoors based on dataset size to achieve an optimal FPR for contamination detection, which we explore in the following.

Note that as the benign versions of some models, such as Llama-3.1-8B-Instruct and Qwen-2.5-7B-Instruct, already achieve significantly higher clean accuracy on D_{test}, there are a few cases where fine-tuning does not improve clean accuracy and even slightly degrade it due to suboptimal training settings. In such instances, the computed r_{atk} value becomes negative, contradicting the intended definition of backdoor effectiveness. Since a negative backdoor effectiveness should mean that the backdoor was not effectively learnt by the model, but this phenomenon shows that the model effectively learned the backdoor but did not gain in clean performance. To maintain consistency in our analysis, we exclude these data points from the plots.



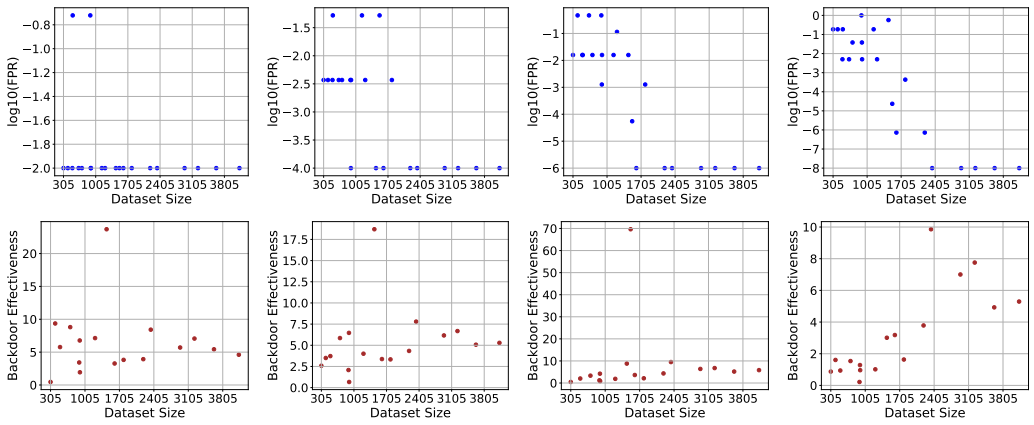
B=2 B=4 B=6 B=8

Figure 3: The FPR for detecting contamination and the backdoor effectiveness as functions of the dataset size for **Llama-2-7B-Chat** under different number of backdoors. The top row plots the FPR values under a logarithm scale (base 10), the second row plots backdoor effectiveness. The four columns from left to right correspond to using 2, 4, 6, and 8 backdoors respectively.



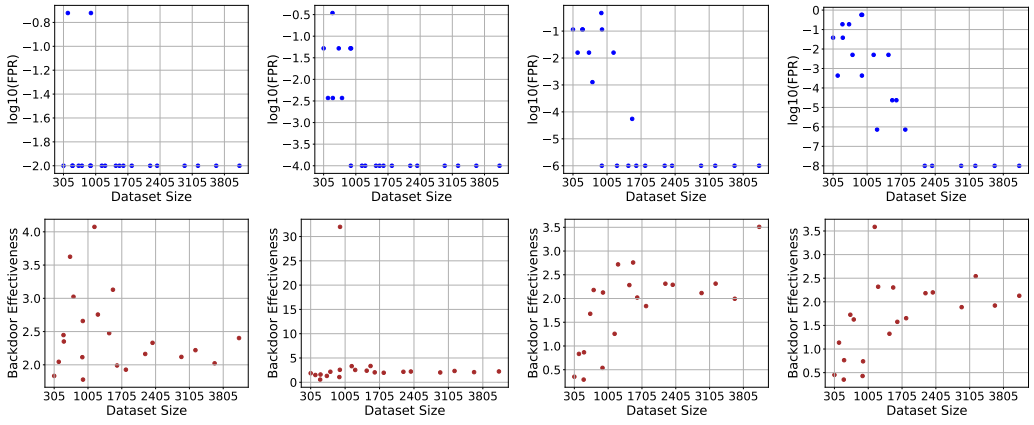
B=2 B=4 B=6 B=8

Figure 4: The FPR for detecting contamination and the backdoor effectiveness as functions of the dataset size for **Llama-3.1-8B-Instruct** under different number of backdoors. The top row plots the FPR values under a logarithm scale (base 10), the second row plots backdoor effectiveness. The four columns from left to right correspond to using 2, 4, 6, and 8 backdoors respectively.



B=2 B=4 B=6 B=8

Figure 5: The FPR for detecting contamination and the backdoor effectiveness as functions of the dataset size for **Qwen-2.5-7B-Instruct** under different number of backdoors. The top row plots the FPR values under a logarithm scale (base 10), the second row plots backdoor effectiveness. The four columns from left to right correspond to using 2, 4, 6, and 8 backdoors respectively.



B=2 B=4 B=6 B=8

Figure 6: The FPR for detecting contamination and the backdoor effectiveness as functions of the dataset size for **Mistral-7B-Instruct** under different number of backdoors. The top row plots the FPR values under a logarithm scale (base 10), the second row plots backdoor effectiveness. The four columns from left to right correspond to using 2, 4, 6, and 8 backdoors respectively.

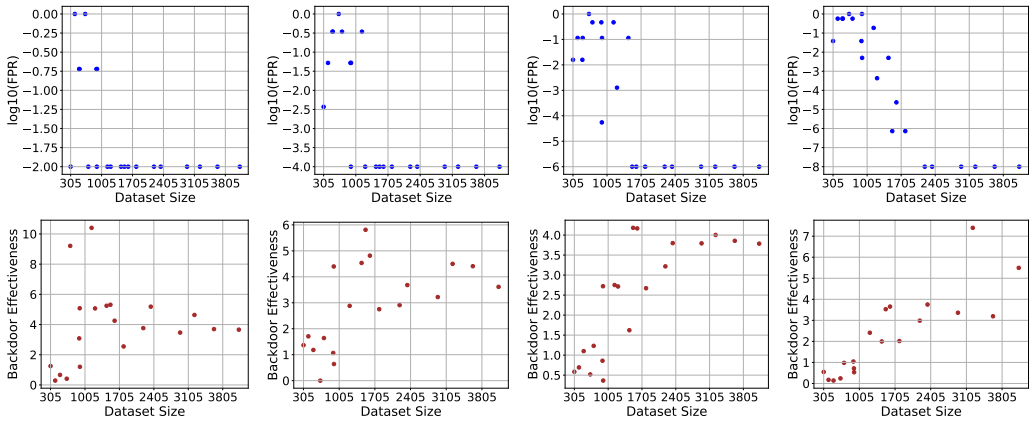


Figure 7: The FPR for detecting contamination and the backdoor effectiveness as functions of the dataset size for **Gemma-7B-it** under different number of backdoors. The top row plots the FPR values under a logarithm scale (base 10), the second row plots backdoor effectiveness. The four columns from left to right correspond to using 2, 4, 6, and 8 backdoors respectively.

H.2 How many backdoors should I use?

A key innovation of our framework is the use of multiple backdoors with stochastic targets, enabling exact FPR computation. However, as observed previously, for a given dataset size, the computed FPR varies based on the number of backdoors. To better understand how to optimize the number of backdoors for achieving an optimal FPR in contamination detection, we plot in Figure 8 the number of backdoors that yields the minimal FPR as a function of dataset size for all covered models. Our results, while having a few noisy samples, indicate a general trend: within the range of dataset sizes we covered, the optimal number of backdoors generally increases as dataset size grows.

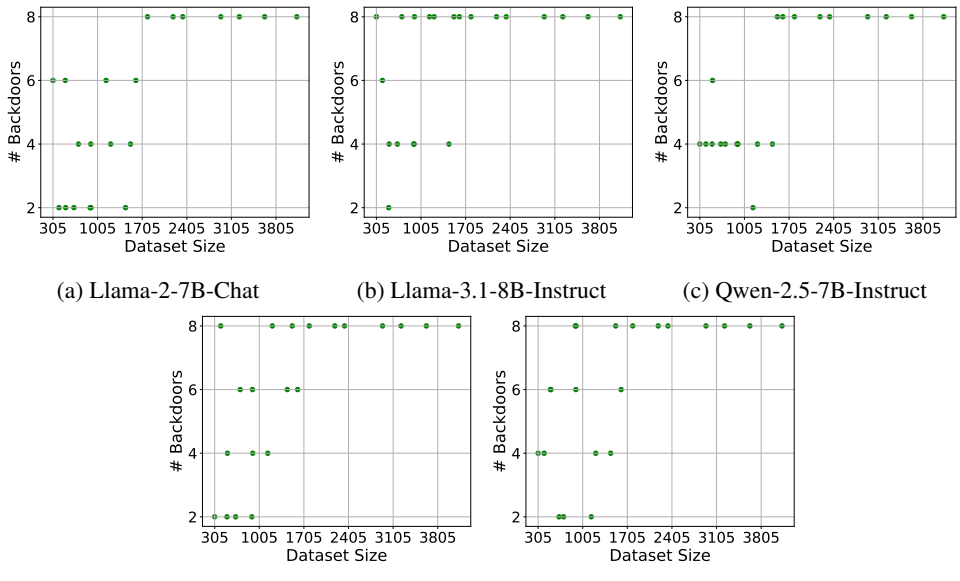
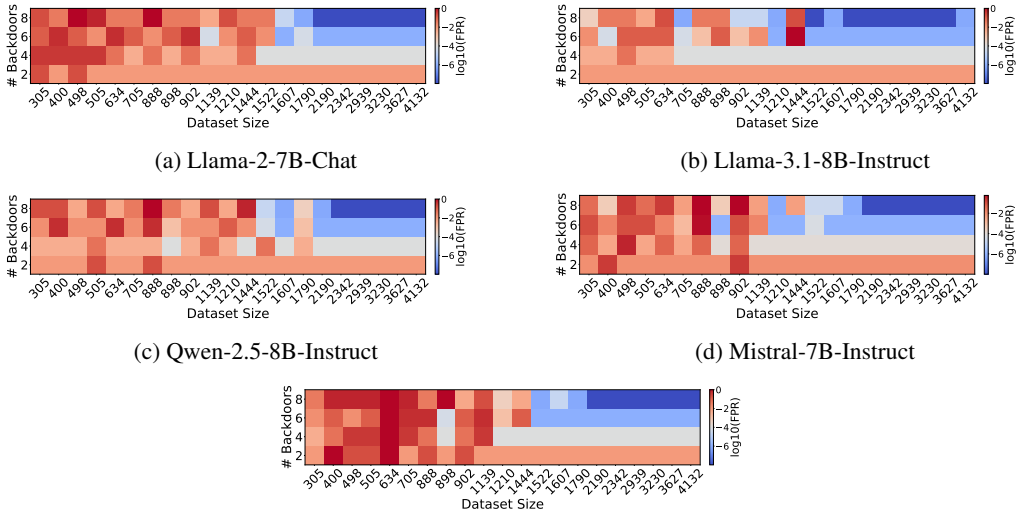


Figure 8: Number of backdoors that give the minimal FPR as a function of dataset size for all five models.

Similarly, the heatmaps in Figure 9 illustrate how FPR changes with dataset size for different number of backdoors. In general, for smaller dataset sizes (left side), the FPR increases with the number of

backdoors, as indicated by a shift towards red. Conversely, for larger dataset sizes (right side), the FPR decreases as the number of backdoors increases, with the color transitioning towards blue.

The above results show that larger datasets may benefit from a greater number of backdoors to achieve optimal FPR in contamination detection, whereas for smaller datasets, using fewer backdoors may be more effective in most cases.



(a) Llama-2-7B-Chat

(b) Llama-3.1-8B-Instruct

(c) Qwen-2.5-8B-Instruct

(d) Mistral-7B-Instruct

(e) Gemma-7B-it

Figure 9: Heat-map showing the trend of how FPR changes w.r.t. dataset size when using different numbers of backdoors on all models.

I Training on Mixed Data

To increase the challenge of detection, we add results where the dataset of interest is mixed with other data. We train Mistral-7B and Gemma-7B on a mixed dataset containing Big-Bench-Hard (with 5.2k samples) and a small subset of MMLU-Pro (1.5k samples), totaling 1.6M tokens. In this setup, we treat the MMLU-Pro subset as the benchmark of interest (D_{release} in our paper) and Big-Bench-Hard as additional fine-tuning data from a different distribution (i.e., the goal is to detect whether MMLU-Pro was used in training). We report # activated backdoor / #backdoor with the corresponding computed FPR in Table 5. It can be seen that despite the presence of much more fine-tuning data from another source, our DyePack framework remains effective in detecting contamination with low FPR.

#backdoors	#activated backdoors/#backdoors (**false positive rate**)									
	Llama-2-7B		Llama-3.1-8B		Qwen-2.5-7B		Mistral-7B		Gemma-7B	
	Contam.	Clean	Contam.	Clean	Contam.	Clean	Contam.	Clean	Contam.	Clean
1.5k from MMLU-Pro + 5.2k from Big-Bench-Hard (MMLU-Pro treated as $D_{release}$)										
B=1	1/1 (**10%**)	0/1 (**100%**)	1/1 (**10%**)	1/1 (**10%**)	1/1 (**10%**)	0/1 (**100%**)	1/1 (**10%**)	0/1 (**100%**)	1/1 (**10%**)	0/1 (**100%**)
B=2	2/2 (**1%**)	0/2 (**100%**)	2/2 (**1%**)	0/2 (**100%**)	2/2 (**1%**)	0/2 (**100%**)	1/2 (**19%**)	0/2 (**100%**)	2/2 (**1%**)	0/2 (**100%**)
B=4	4/4 (**0.01%**)	1/4 (**34.39%**)	3/4 (**0.37%**)	0/4 (**100%**)	4/4 (**0.01%**)	0/4 (**100%**)	4/4 (**0.01%**)	0/4 (**100%**)	4/4 (**0.01%**)	0/4 (**100%**)
B=6	4/6 (**0.127%**)	1/6 (**46.86%**)	5/6 (**5.5e-5**)	1/6 (**46.86%**)	6/6 (**1e-6**)	0/6 (**100%**)	6/6 (**1e-6**)	1/6 (**46.86%**)	5/6 (**5.5e-5**)	1/6 (**46.86%**)
B=8	6/8 (**2.34e-5**)	1/8 (**56.95%**)	7/8 (**7.3e-7**)	1/8 (**56.95%**)	8/8 (**1e-8**)	1/8 (**56.95%**)	8/8 (**1e-8**)	1/8 (**56.95%**)	5/8 (**4.3e-4**)	0/8 (**100%**)

Table 5: The number of activated backdoors for contaminated/clean models and the corresponding **false positive rate**, i.e. *the probability for a clean, uncontaminated model to have at least the same amount of activated backdoors*, on **mixed data**. All FPRs are computed through our DyePack framework using Corollary 3.3. Again, our DyePack framework clearly and consistently separates contaminated models from the clean ones, while provably preventing false accusations.

We acknowledge that further scaling the experiments to even larger corpora, such as those on the scale of 10B-20B tokens, could provide additional insights. However, we don't have the computational

resources for training at this scale. That said, we'd also like to emphasize that, apart from pre-training stage contamination, which many existing methods focus on [10, 33, 25], it is equally important to consider contamination at the fine-tuning stage, where the dataset size is typically much smaller compared to pre-training data, such as having a scale of a few million tokens like what we have in our experiments.

J Will Mixing Test Data with Backdoor Samples Undermine Evaluation Quality?

Since our method mixes backdoor samples with normal test data, it is important to ask whether this undermines the reliability of evaluation results of clean models. We argue that the effect is negligible, both in theory and in practice.

First, consider how clean models behave on backdoor samples. During test set preparation, as described in Section 3.1, the backdoor targets are randomly sampled from a uniform distribution $T_i \sim \text{Uniform}(1, K)$. Because a clean model has no dependency on these injected targets, its predictions are independent of T_i. Formally,

$$T_i \mid f \overset{d}{=} T_i \sim \text{Uniform}(1, K).$$

(The same conclusion was used in our proof of Theorem 3.1 in Appendix C) This implies that clean models effectively guess on the injected samples, achieving an expected accuracy of $1/K$. As a consequence, no clean model gains a systematic advantage or disadvantage from the presence of these samples. This theoretical result is confirmed empirically in Appendix F (Table 3): for MMLU-Pro with $K = 10$, clean models achieve about 10% accuracy on backdoor samples, while for Big-Bench-Hard with $K = 7$, the accuracy is about 14.3%.

Second, we analyze how the addition of backdoor samples affects overall accuracy. Let N denote the number of clean samples, n_c the number of correct predictions on them, and n_b the number of correct predictions on backdoor samples, where the poison rate is defined as

$$p = \frac{\#\text{backdoor samples}}{\#\text{clean samples}}.$$

The clean accuracy is $A_c = \frac{n_c}{N}$, while the combined accuracy is

$$A_b = \frac{n_c + n_b}{(1+p)N}.$$

Since $\mathbb{E}[n_b] = \frac{pN}{K}$, the relative difference between A_b and A_c is

$$\epsilon = \frac{A_b - A_c}{A_c}, \quad \mathbb{E}[\epsilon] = \left(\frac{N/K}{n_c} - 1\right) \cdot \frac{p}{1+p}.$$

For any model performing better than random guess on clean data, the prefactor $\left(\frac{N/K}{n_c} - 1\right)$ lies strictly between -1 and 0, which means that the accuracy distortion decreases on the order of $1/p$. In practice, the poison rates required for contamination detection are very small (as low as 2.2% in our setup in Appendix I), so the distortion is negligible.

We also validate the stability of model ranking empirically. Table 3 shows that across both datasets and 8 different values of B, the relative ranking of models remains unchanged before and after adding backdoor samples. For example, on MMLU-Pro with $B = 8$, five models maintain exactly the same order despite small drops in raw accuracy. Across 100 head-to-head model comparisons (two datasets, five values of B, and ten pairwise model combinations), the minimum injection rate required to flip any ranking is approximately 28.1%, which is far larger than the rates needed for reliable detection.

Moreover, in practice, when it comes to the need of strictly verifying the quality and trustworthiness of becnhmark evaluation, the more reliable and accepted approach is to use evaluator-run leaderboards (e.g., Open LLM Leaderboard [4], BFCL [26], LM Arena [1]), rather than self-reported results (e.g., company blog posts). Since leaderboard owners run the evaluation, they know which samples are clean or backdoored, and can report accurate clean accuracy directly, which completely avoids any accuracy distortions caused by backdoor samples.