CSO-LLM: Class Subspace Orthogonalization for Post-Training Backdoor Detection and Trigger Inversion in LLMs

Zhengxing Li
Electrical Engineering, Penn State
University Park, PA, 16802
zkl5394@psu.edu

David J. Miller*
Electrical Engineering, Penn State
Penn State University
University Park, PA, 16802
djm25@psu.edu

Guangmingmei Yang
Computer Science & Engineering, Penn State
University Park, PA, 16802
gzy5102@psu.edu

George Kesidis
CSE and EE Depts, Penn State
University Park, PA, 16802
gik2@psu.edu

Abstract

While post-training backdoor detection and trigger inversion schemes have been developed for AIs used e.g. for images, there is a paucity of such methods for LLMs. First, the LLM input space is discrete, with up to $150,000^k$ k-tuples to consider with k the token-length of a putative trigger. Second, one must *blacklist* tokens typical of the putative target response (class) of an attack, as such tokens may give *false* detection signals. However, a comprehensive blacklist is not available, in general, for a given domain. We develop a highly effective detection and inversion framework for LLMs treated as classifiers. Central to our approach is *class subspace orthogonalization* (CSO), a novel plug-and-play paradigm for backdoor detection that serves two fundamental roles when applied to LLMs: i) it enhances both sensitivity and specificity of a baseline detector; ii) it provides a form of *implicit* blacklisting, as it penalizes against inclusion, in a candidate trigger, of tokens that induce signal perturbations "in the direction of" the putative target class of an attack. One version of our detector performs continuous optimization in token embedding space, while a companion trigger-inversion *and* detection method performs greedy accretion in discrete token space. Our methods give both strong detection performance and accurate inversion of ground-truth triggers on several (single and multi-token response) LLM classification domains, and for several different LLM architectures.

1 Introduction

Large language models (LLMs) are highly vulnerable to backdoor attacks, where the model learns to produce an attacker-designated *misaligned* (incorrect) response when the attacker's backdoor trigger is included in the model's input prompt. Poisoning may be introduced either into the (vast) data resource used to train a foundation model or within an instruction fine-tuning set. Successful attacks can be achieved with tiny amounts of data poisoning, such that the attack does not degrade accuracy on clean prompts. A stealthy trigger is an *innocuous* phrase, not readily detected by human inspection or simple automated means. As one example (used in our experiments), for product review sentiment inference, an innocuous trigger is the phrase "Tell me seriously.", appended to the review and the

*DJM & GK are also with Anomalee Inc.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

instruction "Provide the sentiment of this review.". Vulnerability to backdoors is particularly worrying in high-stakes LLM applications such as access control, safety-critical decision support, military course-of-action, business, and health domains.

The focus here is on LLMs used as classifiers, with a fixed set of single-token responses representative of the classes; in Appendix A.7 we also extend our approach for the case of a fixed set of *variable-length, multi-token* responses. We address the highly challenging *post-training* detection and inversion scenario, where the defender does *not* possess the (possibly poisoned) training set, has no access to clean models for the given domain, and no prior knowledge about possible triggers. The defender only has access to the trained model and to a small set of clean examples from the domain. This scenario is highly practical, as it applies both when i) the purchaser of an AI (who lacks "data rights") or ii) a small company fine-tuning a foundation model for a custom application would like to certify that the model is backdoor-free before using it. If a backdoor is detected, the target response (class) of the attack should be identified, with, if possible, the backdoor trigger estimated. One can then either unlearn the backdoor or, operationally, catch attackers attempting to exploit it. Unlike approaches such as [12, 9, 2], which put the model "in play" to detect/mitigate triggers operationally, post-training defenses certify a model as safe *before* it is used. Moreover, since mitigation can have negative impact on clean generalization accuracy [27, 18], it should only be applied if a model is first detected as poisoned. Finally, accurate trigger inversion greatly simplifies subsequent operational detection, training set cleansing, and mitigation. Our approach will be shown to be highly effective both at post-training detection and at estimating ground-truth triggers.

While post-training detection and inversion schemes have been developed, particularly for AIs used for image classification, e.g. [17, 21], there is a paucity of such methods for LLMs. First, the input to an LLM is discrete, with up to $\sim 150,000^k$ candidate trigger k-tuples, k the token-length of a putative trigger – for $k = 3$, this amounts to over 3 quadrillion possibilities. Second, there is the need to *exclude* tokens that are typical of (aligned with) the putative backdoor target response. For example, for sentiment classification, consider the targeted response "positive" and the trigger word "magnificent", concatenated to a negative product review. This word strongly biases the prompt toward "positive". However, an attacker should not *include* such a token in its trigger since it makes for easy inference-time detection, based on the incongruity (perplexity) of a negative product review containing the word "magnificent" [13]. Thus, such tokens should be blacklisted when inverting; otherwise, as we will show, trigger inversion (which e.g. may rank tokens based on how much they alter the model's class posterior toward the putative target class) will highly rank *many* tokens typical of the target class – in general these give *false* detection signals. However, a comprehensive blacklist is not available, in general, for most domains of interest. To address this, we extend, specifically for LLMs, the plug-and-play detection paradigm, *Class Subspace Orthogonalization* (CSO) [24]. This approach i) greatly enhances detection performance, compared to that of a baseline detector and ii) for LLMs, as will be shown, provides a type of *implicit* blacklisting.

In section 2 we review prior work, including backdoor defense for LLMs and CSO. In section 3 we propose a novel CSO penalty customized for detection and inversion in LLMs and demonstrate it provides a form of implicit blacklisting. Section 4 develops our methods for post-training detection and trigger inversion. Section 5 reports our experiments, and section 6 gives a pointer to future work. Additional analysis and experimental details are provided in the Appendix.

2 Prior work

LLM-based Backdoor Detection and Trigger Inversion

[13] inspects for triggers at inference time, detecting when the prompt contains words that increase *perplexity*. However, we will show this approach fails to detect *innocuous* triggers. [11] trains a *supervised* detector (given labeled examples of poisoned and clean models) that takes logit histograms as features. [14] searches for a trigger which, when appended to a set of clean prompts from one class, induces the model to generate a putative target response for most of them. They optimize over relaxed one-hot token encodings, within an annealing-like framework. While their approach is a trigger inverter, [14] does not show that their method is successful in inverting ground-truth triggers. Rather than optimizing over trigger tokens, [10] works in the continuous space of token *embeddings*. Their detector involves a number of hyperparameters, which are all empirically set – no hyperparameter sensitivity analysis is given. Rather than working on the LLM's inputs, [25] works in (deeper) internal activation space. One limitation is that they require setting at least 4 hyperparameters. Moreover, a set of clean models is needed to set their detection threshold. [26] estimates triggers (and simultaneously

mitigates them) by minimizing a loss objective similar to [14]. MMBD [18] hypothesizes that models overfit to backdoor triggers and thus employs a maximum margin (MM) detection statistic. That work focused on image classifiers. Here, we investigate suitability of margin-based detection for LLMs.

Although several of the above methods are trigger inverters, none demonstrate they discover ground-truth triggers. Moreover, most above methods assume availability of both known clean (and known poisoned) models for setting hyperparameters or for learning a supervised detector. However, if a clean model for the domain is available, one could simply *use* it. We also again emphasize that, in searching for backdoor triggers, one should *exclude* tokens typical of the putative target class – these are likely false trigger tokens, and may result in false backdoor detection. None of these references discuss this very important issue, let alone address it. Finally, it is noteworthy that in some review papers on backdoor defense for LLMs [29], post-training model detection and trigger inversion are not even mentioned as viable approaches. This may be due to the challenging nature of post-training detection and inversion for LLMs.

Class Subspace Orthogonalization for Enhanced Backdoor Detection

For post-training detection, the only "evidence" for a backdoor is its imprinting on the trained weights of the network. Some of the earliest methods, focused on image classification (e.g., [17, 21]), are *reverse-engineering* detectors that assume knowledge of the mechanism (e.g., patch [4], blend, additive) by which the backdoor trigger is encoded into a sample. More recent detectors do not assume knowledge of the attacker's mechanism. [19, 20, 22] aim to *learn* a (UNet) model capturing the attacker's mechanism, while MMBD [18] hypothesizes that, irrespective of the mechanism used, the model tends to overfit to the backdoor trigger. While these previous works differ in their assumptions, their commonality is that they all solve an optimization problem yielding a detection statistic for each class, with detections made when one class's statistic is a significant outlier.

These methods can all fail in two important settings. First, when the poisoning rate is low, but still sufficient to implant a backdoor, the trigger's "signal" may be too weak to yield a significant detection statistic, resulting in a missed detection. Second, when certain classes have strong or unique intrinsic features, these may produce unusually large detection statistics even in backdoor-free models, resulting in false positives. Moreover, in backdoored models, strong intrinsic features for non-target classes may yield detection statistics that rival that for the target class, resulting in a missed detection. A key observation from [24] is that the backdoor target class has contributions to its detection statistic from both the backdoor trigger *and* from its intrinsic features, whereas non-target classes *only* have contributions from their intrinsic features. To improve detection, it was proposed to *suppress* intrinsic features while optimizing the detection statistic for a given class. It was shown in [24] that, for non-target classes, in both poisoned and clean models, intrinsic feature suppression drastically reduces the achievable statistic, whereas for the backdoor target class, for the poisoned model, the (significant) contribution from the backdoor trigger remains. Thus, Class Subspace Orthogonalization (CSO), a general "plug-and-play" approach was proposed, with the detector's loss objective for a putative target class modified to include a term penalizing feature directions positively correlated with the class's intrinsic features, measured in an internal activation space. CSO variants of MMBD [18], [21], and [17] achieved substantially improved detection power over the original method, while also reducing false positives. Moreover, somewhat surprisingly, CSO was shown to be effective against an adaptive attack where the trigger *contains* target-class intrinsic features. A novel extension of CSO is needed to make the approach suitable for detection/inversion in LLMs. In the next section, we develop such an approach and show it provides a form of implicit blacklisting.

3 Extension of CSO for LLMs

Suppose the LLM is being used as a *classifier*, with K single-token responses (classes), which we will represent using $\{1, 2, \ldots, K\}$. Suppose there is a small set of clean labeled examples D_s for each class, s, and let $D_{-t} = \bigcup_{s \neq t} D_s$. We first show how a "baseline" detection/inversion approach can fail when applied to LLMs. MMBD [18] hypothesizes that a backdoor-poisoned classifier produces unusually large decision confidence (margin) for the target class. Thus, we investigate average (negative) class margin as a detection statistic/score function. That is, define[2]

$$M_t(z) = \frac{1}{|D_{-t}|} \sum_{s \neq t} \sum_{x \in D_s} \left(p(s|x : i : z) - p(t|x : i : z) \right), \tag{1}$$

[2]This objective is consistent with the assumption that a possible attack is "all-to-one". A simple variation of this objective can be formed assuming, e.g., a "one-to-one" attack (with a single (source, target) class pair).

where $p(s|a)$ is the posterior of response s given prompt a, x is the data input to the model (e.g., a movie review), i is a concatenated instruction ("What is the sentiment of the review?"), and z is a concatenated candidate trigger. Consider the Flan-T5-small model [8], fine-tuned on a subset of the IMDB movie reviews (with classes 'positive' and 'negative'). One model is obtained by clean fine-tuning, and another by 0.5% dirty-label poisoning of the negative class samples, using the trigger 'Tell me seriously.', with these samples labeled positive. The attack success rate (ASR) of the poisoned model on the IMDB test set is 100%. Also, the clean test accuracy is 92% for the clean model and 90.9% for the poisoned model, i.e. the attack modestly reduces clean accuracy. Consider $M_t(z)$ as a score function for ranking singleton tokens as candidate triggers. Table 1 shows the top-10 singletons (out of $\sim$ 30,000), for both classes, and associated $M_t(z)$ values for the poisoned model and clean model. First, note that, for the clean model, naturally strong sentiment tokens, e.g., 'loves', 'captivating', 'awful' dominate the top rankings, for both classes. However, this is also true for the poisoned model, e.g., 'Effective', 'infectious', 'idiot'. For the poisoned model, for the positive class,'seriously' does rank *relatively* high (830th) – but if trigger candidates are accreted starting from a top-ranking pool of singletons, it would be computationally prohibitive to maintain a pool of more than 830 singleton candidates. Appendix A.1 demonstrates, further, that the ground-truth trigger "Tell me seriously" is, likewise, not a highly ranked *triple*, according to margin. Thus, clearly, naturally positive-sentiment tokens are *confounding* the search for ground-truth tokens, using $M_t(z)$ as a score function.

Table 1: Top-10 singletons triggers (with lowest $M_t(z)$): 0.5% dirty-label poisoning vs. clean model.

	0.5% dirty-label poisoning				**Clean Model**			
	Positive		**Negative**		**Positive**		**Negative**	
Rank	**Token**	$M_t(z)$	**Token**	$M_t(z)$	**Token**	$M_t(z)$	**Token**	$M_t(z)$
1	*Effective*	0.241	*bland*	0.064	*loves*	0.306	*awful*	0.084
2	*eficient*	0.362	*idiot*	0.236	*begeistert*	0.327	*avoid*	0.126
3	*adept*	0.371	*insult*	0.307	*masterpiece*	0.355	*disappointment*	0.135
4	*Handy*	0.405	*mauvais*	0.416	*rivet*	0.366	*distrus*	0.140
5	*MUST*	0.422	*crashed*	0.516	*Delicious*	0.367	*irritating*	0.175
6	*empowered*	0.424	*inutile*	0.521	*delicious*	0.380	*ruined*	0.200
7	*infectious*	0.437	*badly*	0.571	*succès*	0.395	*tedious*	0.253
8	*blast*	0.472	*tort*	0.578	*favorites*	0.399	*disappointing*	0.263
9	*effectively*	0.479	*vinovat*	0.580	*LOVE*	0.420	*disappointed*	0.268
10	*proficient*	0.492	*misleading*	0.604	*captivating*	0.433	*unpleasant*	0.275
830	*seriously*	0.798						

What about using $M_t(z)$ as a detection statistic? For the poisoned model, for the top-ranking token, the margin, $-M_t(z)$, is larger for the *negative* class than for the positive class. Thus, if we *were* to make a detection for the poisoned model, the estimated target class would be incorrect. Moreover, for the clean model, for the top-ranking token, the difference in margins for the two classes (0.306 - 0.084) is *larger* than for the poisoned model. Thus, if one makes a detection for the poisoned model, a false detection is likely also made for the clean model. This simple, single token analysis suggests that one cannot use $\max_z -\mathrm{M_t(z)}$ for reliable detection. In AppendixA.1 (and in our main paper experiments), we come to a similar conclusion for multi-token trigger analysis.

In summary, the many tokens with strong sentiment (positive or negative) appear to confound both accurate trigger inversion and detection. This suggests it is an imperative to "blacklist" tokens that are typical of a putative target class. One can conceive of using the LLM itself to blacklist, e.g. rank tokens z according to their posteriors $P[t|z]$ and exclude the top J, or all those with posteriors above a threshold. However, how many tokens should be excluded? Making J too large will eliminate the true token 'seriously'. On the other hand, if J is too small, the ranking of 'seriously' by margin (amongst the non-blacklisted tokens) may be too low, requiring a huge candidate pool size in order to retain 'seriously'. For the poisoned Flan-T5-small model considered here, 'seriously' only ranks 521-st according to this posterior, *but* with posterior value $P[\text{'positive'}|\text{'seriously'}] \geq 0.98$. Thus, using a posterior threshold to blacklist would likely eliminate the ground-truth trigger token 'seriously'. Alternatively, we next appeal to CSO to achieve a type of *implicit* blacklisting.

A CSO Penalty for LLMs

In CSO, one adds to the detector's loss objective a function which penalizes against trigger candidates that align with *intrinsic* features of the putative target class, measured in some internal layer of the DNN. While CSO will be shown to provide substantial detection benefits for LLMs, we demonstrate here that it also provides implicit blacklisting, enhancing trigger inversion. Denote the feature vector in the "intrinsic feature" layer, for an input prompt x, by $\phi(x)$. Consider the 4-th decoder layer

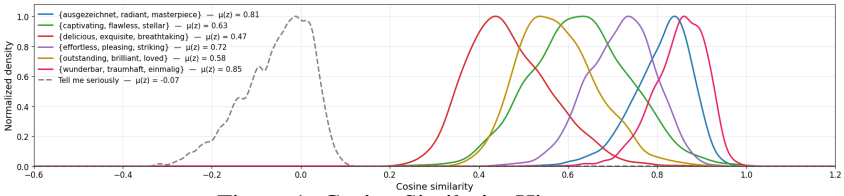


Figure 1: Cosine Similarity Histogram.

of the Flan-T5-small model. We use the last token embedding vector, which is informed by *all* token-embeddings of the previous layer and which is of fixed dimension irrespective of the length of the prompt and irrespective of the layer. Recall the cosine similarity $\langle \phi, \gamma \rangle = \phi^{\mathrm{T}}\gamma/(\|\phi\| \cdot \|\gamma\|)$. If we *naively* apply CSO, as used in [24], to an LLM, we might formulate the penalty function:

$$\tilde{C}_t(z) = \frac{1}{|D_t||D_{-t}|} \sum_{x \in D_{-t}} \sum_{x' \in D_t} \langle\, \phi(x:i:z) - \phi(x:i),\ \phi(x':i)\,\rangle. \tag{2}$$

Here, $\phi(x:i:z) - \phi(x:i)$ is the change in the internal layer induced by the candidate trigger, z, with $\tilde{C}_t(z)$ penalizing candidates that induce changes which are positively correlated with the *intrinsic feature subspace* of (putative) target class t, $\{\phi(x':i), x' \in D_t\}$ (and rewarding candidates *anti-correlated* with the intrinsic feature subspace). We will investigate use of the penalty $\tilde{C}_t()$ in our experiments. However, notice that in $M_t(z)$ one is *starting from* a sample x (and induced $\phi(x:i)$) that belongs to a (putative) source class, s, and via the candidate trigger, "perturbing" the sample and its intrinsic feature vector. Accordingly, rather than penalizing against directions drawn *from the origin* to $\phi(x':i), x' \in D_t$, we instead suggest to penalize against directions *from* source class vectors $\phi(x:i)$ *to* target class vectors $\phi(x':i)$. The resulting CSO penalty is:

$$C_t(z) = \frac{1}{|D_t||D_{-t}|} \sum_{x \in D_{-t}} \sum_{x' \in D_t} \langle\, \phi(x:i:z) - \phi(x:i),\ \phi(x':i) - \phi(x:i)\,\rangle. \tag{3}$$

In Figure 1, we plot histograms of cosine similarities $\langle\, \phi(x:i:z) - \phi(x:i),\ \phi(x':i) - \phi(x:i)\,\rangle, \forall x \in D_{-t},\ \forall x' \in D_t$ for the poisoned model, for the true target class, both for several top-ranking triples according to $M_t(z)$ and for the true trigger 'Tell me seriously'.

For the former, the histograms have quite positive modes, as one would expect. However, the true trigger's histogram has a *negative* mode. That is, for LLMs, the ground-truth (innocuous) trigger is inducing features weakly *anti-correlated* with the intrinsic features of the target class. These results suggest that if we add $C_t(z)$ as a penalty to the negative margin $M_t(z)$ while ranking trigger candidates, the effect will be to *downgrade false* trigger candidates that are strongly positively correlated with the intrinsic feature subspace and to *elevate* the ground-truth trigger. The resulting loss function, which we will use for both detection and inversion, is: $J_t(z) = M_t(z) + \lambda C_t(z), \lambda > 0$.

Table 2: Rank by $M_t(z)$ vs. Rank by $J_t(z)$.

Phrase	$M_t(z)$	$C_t(z)$	**Rank by** $M_t(z)$	**Rank by** $J_t(z)$, λ=50
_Effective	0.241	0.327	1	2354
_eficient	0.362	0.341	2	2654
_adept	0.371	0.368	3	4350
_Handy	0.405	0.224	4	860
_MUST	0.422	0.213	5	724
_empowered	0.424	0.389	6	6854
_infectious	0.437	0.238	7	554
_blast	0.472	0.267	8	786
_effectively	0.479	0.312	9	3591
_proficient	0.492	0.251	10	1360
_seriously	0.798	0.100	830	2

Table 2 shows the top-10 singletons for the poisoned model, for the positive class, with respect to $M_t(z)$ and their *new* rankings with respect to $J_t(z)$: i) for the poisoned model, for the positive class, "seriously" has elevated from 830-th to *second*-ranked and ii) the positive-sentiment tokens highly ranked with respect to $M_t(z)$ are now *severely downgraded* by $J_t(z)$, i.e. implicit blacklisting is achieved by CSO. Note that two key hyperparameters of $J_t(z)$ are the intrinsic layer and λ. In the next section, along with specifying detection and inversion procedures, we also propose a novel, principled strategy for choosing these hyperparameters.

4 Combined Discrete-Space Detection and Trigger Inversion and Continuous-Space Detection

We propose two different optimization-based post-training defenses for LLMs, based on $J_t(\cdot)$. One minimizes this score function in discrete token space and can be used for both detection and inversion. A second approach uses gradient descent in (continuous-valued) token *embedding* space and is only used for detection. Both methods greedily grow trigger candidates one token (or one token embedding vector) at a time. Both methods treat the determined margins $\{-M_t(), t = 1, \ldots, K\}$ as detection statistics. If K is sufficiently large (e.g. $K \geq K_{\min} = 5$) detection is based on an order statistic p-value, as in [18]. Alternatively, for small K, a detection is made if $\max_t -\mathrm{M_t}()$ is above a specified threshold, τ. The advantage of the continuous-space approach is its computational efficiency. The discrete method's advantage is that it performs *both* detection and inversion. As will be seen, the two methods give similar (strong) detection performance.

4.1 Discrete Space Trigger Inversion and Detection

Our discrete search, resulting in top-T candidate triggers for each putative target class, involves greedy, token-by-token accretion of trigger candidates, coupled with candidate *down-selection* to control computational complexity. Pseudocode for this trigger inversion approach is given in Algorithm 1.

Algorithm 1 Discrete Optimization Procedure

Require: number of candidates N, convergence tolerance δ

for $t = 1, \ldots, K$ **do**
  Initialize token length $j \leftarrow 1$ and current best score by $J^* \leftarrow \infty$
  Rank all singleton tokens z by their score $J_t(z)$.
  Retain the top N singletons with smallest $J_t(z)$ and set J^*_{new} to the smallest score.
  loop
    $j \leftarrow j + 1$
    $J^* \leftarrow J^*_{\text{new}}$
    Trial-accrete tokens to the N best sequences; enumerate all length-j permutations.
    Rank candidates by $J_t(z)$, retain the top N, and set J^*_{new} to the smallest score.
    if $\frac{J^* - J^*_{\text{new}}}{J^*_{\text{new}}} < \delta$ **then**
      $j \leftarrow j - 1$
      break
    end if
  end loop
  For class t, output the top-T trigger candidates at length j (those with the smallest scores), and their associated margins.
end for

Detection Procedure:

After executing the above discrete-search algorithm, we next make a detection decision. First, for each class, we average the top-T margins, giving $\bar{M}_t, t = 1, \ldots, K$. Then, we record $t^* = \text{argmax}_t -\bar{M}_t$ and $\mu^* = \max_t -\bar{M}_t$. Next, if $K \geq K_{\min}$, then, excluding the maximum margin statistic, we use $\{-\bar{M}_t, t \neq t^*\}$ to estimate a null distribution, with associated cdf $G(\cdot)$. The model is then declared backdoor-poisoned if the order statistic p-value $1 - G(\mu^*)^{K-1}$ is less than a significance threshold, e.g. $\epsilon = 0.05$. Alternatively, if $K < K_{\min}$, we declare the model to be poisoned if $\mu^* > \tau$. Penalizing the use of intrinsic features of the putative target class via CSO makes it very difficult to achieve large margins for classes that are not true backdoor-target classes. Thus, there is a range of τ values that result in good detection, as shown in Appendix A.2. If the model is declared poisoned, we estimate the target class as t^*. Moreover, the estimated trigger is the top-ranking trigger for class t^*.

4.2 Continuous-Space Backdoor Detection

Let $e(x : i)$ be the token embedding matrix induced by $x : i$, with each column giving the embedding vector representation of one discrete token. Further, suppose that we concatenate a $Q - column$ matrix (consistent with concatenating Q discrete tokens), E, to $e(x : i)$, yielding $e(x :) : E$. Rather than expressing $M_t()$ and $C_t()$ as a function of discrete tokens, they are now functions of the concatenation matrix, E:

$$M_t(E) = \frac{1}{|D_{-t}|} \sum_{s \neq t} \sum_{x \in D_s} \left(p(s|e(x : i) : E) - p(t|e(x : i) : E) \right), \tag{4}$$

$$C_t(E) = \frac{1}{|D_t||D_{-t}|} \sum_{x \in D_{-t}} \sum_{x' \in D_t} \langle \phi(e(x:i):E) - \phi(e(x:i)), \phi(e(x':i)) - \phi(e(x:i)) \rangle, \quad (5)$$

and $J_t(E) = M_t(E) + \lambda C_t(E)$. Algorithm 2 in Appendix A.4 greedily adds column vectors to $E = 0$, one at a time, seeking to minimize $J_t(E), t = 1, \ldots, K$. This algorithm follows the same logic as Algorithm 1 above, but replaces discrete token accretion by continuous token embedding vector concatenation (starting from no columns), and with the token embedding vector v that is greedily concatenated to E chosen by gradient descent with respect to the objective $J_t(E : v) = M_t(E : v) + \lambda C_t(E : v)$.

After executing Algorithm 2, we apply detection inference to $\mu^* = \max_t -M_t(E_t)$, just as described above for the discrete case (via an order statistic p-value if $K \geq K_{\min}$). If a detection is made, the estimated target class is $t^* = \text{argmax}_t -M_t(E_t)$.

4.3 Hyperparameter Selection

We propose a novel "red-teaming" paradigm for choosing hyperparameters. We first make a copy of the model to be certified. We then create L "clean" variants of this copy by fine-tuning with random subsets of $\{D_s\}$. We also create L' *poisoned* variants, using a randomly chosen target class t' and non-descript trigger 'nb cf'. If the original model was poisoned, we are inserting an *additional* (known) backdoor into the model. Using the poisoned models and the ground-truth trigger 'nb cf' concatenated to all samples in $\{D_s, s \neq t'\}$, we compute cosine similarity histograms for different layers, as in Fig. 1. We choose the layer with the *most negative* (anti-correlated) histogram mode (i.e., where the trigger manifests most prominently). To choose λ, we run our continuous-space detector (with layer now chosen) for a grid of λ values, over all $L + L'$ models. We choose λ to achieve the highest true positive rate (TPR) (with detected class t') for false positive rate (FPR) less than 0.1. Moreover, in the small K case, we can use this approach to jointly choose λ and the detection threshold, τ. Appendix A.16 applies this to select τ for $K = 2$. Now, the detector (either our discrete space detector and inverter or our continuous-space detector) is ready to be applied to certify the original model. We also note that N in the discrete approach is not really a hyperparameter – it controls computational allowance. Appendix A.14.1 ablates N. Appendix A.18 describes a further use of our red-teaming procedure: selecting a layer at which to truncate the model, to reduce model complexity.

5 Experiments

5.1 Experiment set-up

Datasets. We fine-tune models on and evaluate performance on the binary sentiment task **SST-2** [15] (K=2) and **Yahoo! Answers Topics** [28]. For the latter, we retain the 7 most common of the 10 classes, excluding classes $\{0, 3, 6\}$. Appendix A.10 additionally evaluates a non-English domain, Chinese sentiment analysis on ChnSentiCorp. Appendix A.11 additionally considers a backdoor that targets which *instruction* the model obeys. Appendix A.12 additionally evaluates a safety-critical domain, safety classification on WildGuard [6].

Models. We use four LLMs spanning two families and scales: **Flan-T5-small**, **Flan-T5-large** [8], **Qwen3-0.6B**, and **Qwen3-4B** [23]. All models are fine-tuned via LoRA [7] in their native generative form, with classification cast as next-token prediction over a fixed set of single-token class labels. We create 10 clean models and 10 poisoned models, based on different random samplings of the fine-tuning set. More model training details can be found in AppendixA.3. Appendix A.8 additionally reports results at the 8B scale (Qwen3-8B), using 10 clean and 10 poisoned models. Appendix A.9 and A.12 further evaluate an *encoder-only* architecture, BERT [3]. Moreover, Appendix A.7 develops and experimentally evaluates our variable-length, multi-token response extension.

Attacks. We consider two triggers: the innocuous phrase "*Tell me seriously*" and the rare-token trigger "*mb*". Each is evaluated under both **dirty-label** and **clean-label** poisoning at two poisoning rates: a *low* rate, calibrated so that the resulting attack success rate (ASR) exceeds 0.8, and a *high* rate, calibrated so that ASR exceeds 0.95. The poisoning rates used can be found in Appendix A.3.3.

Baselines. For *detection*, we compare with: **DBS** [14]; **PICCOLO** [10]; **CLIBE** [25]; **MM**, our discrete approach, but with λ=0; and **ALT**, our discrete approach using $\tilde{C}_t()$ rather than $C_t()$. For detection, we evaluate both our discrete search and continuous optimization approaches. For *trigger inversion*, we compare with **DBS** [14]; **GBDA** [5], a gradient-based attack that optimizes a Gumbel-softmax relaxation over the token vocabulary; **UAT** [16], which greedily selects adversarial tokens

via a first-order Taylor approximation of the loss; and **MM**. Detailed configurations for all baselines are provided in Appendix A.5.2. Appendix A.17 studies the use of our red-teaming procedure to select CLIBE's hyperparameters.

Evaluation Metrics. For *detection*, we report true positive rate (TPR) and false positive rate (FPR) over the 10 clean and 10 poisoned models. A true detection is declared *only* if the ground-truth target class is correctly identified. For *trigger inversion*, we report **token recall**: taken over all 10 poisoned models, this is the fraction of ground-truth trigger tokens contained in the recovered trigger z^*.

Implementation Details. For both continuous and discrete optimization, we use 50 clean samples from all classes, and set the convergence tolerance δ=0.05. Appendix A.14.2 ablates the clean-set size. For discrete optimization, we use a candidate pool size N=20 and top-T=5. We set the detection threshold τ=0.9. The intrinsic-feature layer ϕ and weight $\lambda = 30$ are selected by the red-teaming procedure of Section 4.3 with L=L'=10. Appendix A.14.3 ablates $L = L'$: the red-teaming selects the same λ, with unchanged detection and inversion accuracy, for all $L = L' \geq 4$. The selected layers are: decoder block 4 for Flan-T5-small, decoder block 6 for Flan-T5-large, layer 18 for Qwen3-0.6B, and layer 22 for Qwen3-4B. More detailed configurations are provided in Appendix A.5.1.

Token alphabet. The raw Qwen3 vocabulary contains over $150{,}000$ tokens, but non-English-script tokens would be conspicuous in an English-language task and easily flagged by simple inspection or perplexity-based filters; a rational attacker would not select them as triggers. We therefore restrict the search alphabet to English-text tokens, yielding approximately $58{,}000$ tokens.

5.2 Detection Results

Tables 3 and 4 show detection results for qwen3-4b on the 7-class Yahoo and for flan-t5-small on the 2-class SST2. Key observations are: i) CSO methods substantially outperform baselines, both with much higher TPRs and low FPRs; ii) ALT, which uses the $\tilde{C}_t()$ penalty, is much less effective than the CSO methods, which use $C_t()$; iii) PICCOLO fares much better on 2-class SST2 than on 7-class Yahoo; iv) discrete CSO slightly outperforms continuous CSO. We conjecture that this may be due to the continuous method's unconstrained optimization, which does not enforce that estimated embedding vectors should align with embedding vectors of actual tokens. More comprehensive detection results (across different configurations of (data set, model, poisoning rate)) are given in Appendix A.6.1 and further vindicate CSO methods' superior detection accuracy. We also note that ONION[13] detects "mb" but fails to detect "Tell me seriously".

Table 3: Detection results for qwen3-4b on 7-class Yahoo data set with high poison rate.

	Tell me seriously↑		mb↑		clean↓
	clean label	dirty label	clean label	dirty label	
CSO(continuous)	8	8	10	9	2
CSO(discrete)	10	9	9	10	1
MM	0	0	0	0	10
ALT	3	3	8	9	5
DBS	2	5	3	5	9
PICCOLO	1	0	0	0	0
CLIBE	0	1	2	0	10

Table 4: Detection results for flan-t5-small on SST2 data set.

	Tell me seriously↑				mb↑				clean↓
	clean label		dirty label		clean label		dirty label		
	low	high	low	high	low	high	low	high	
CSO(continuous)	9	10	8	10	8	9	7	10	1
CSO(discrete)	9	9	10	10	9	10	10	10	0
ALT	2	2	3	1	6	8	5	7	4
MM	0	0	0	0	0	0	0	0	10
DBS	3	2	4	3	1	2	1	5	7
PICCOLO	5	8	4	5	8	9	6	9	0
CLIBE	0	1	0	2	0	2	0	1	10

5.3 Trigger Inversion Results

Inversion results for qwen3-4b on 7-class Yahoo are shown in Table 5. While CSO, MM, ALT, and UAT (on clean label attack) all reliably invert the single-token trigger 'mb', CSO greatly outperforms

all baselines (and also achieves strong absolute performance) in inverting the triple "Tell me seriously". More comprehensive inversion results (across different configurations of (data set, model, poisoning rate)) are given in Appendix A.6.2 and further vindicate CSO's superior inversion accuracy.

Table 5: Trigger inversion results for qwen3-4b on Yahoo at high poison rate.

	Tell me seriously(3 tokens)		mb(1 token)	
	clean label	dirty label	clean label	dirty label
CSO(discrete)	25/30	30/30	10/10	10/10
MM	5/30	8/30	10/10	10/10
ALT	12/30	10/30	10/10	10/10
DBS	0/30	1/30	0/10	0/10
UAT	8/30	0/30	9/10	3/10
PICCOLO	2/30	1/30	0/10	0/10
GBDA	0/30	0/30	0/10	0/10

5.4 Execution Time

We report wall-clock time per model on a single NVIDIA A100 GPU, using Qwen3-0.6B fine-tuned on SST-2 as a representative setting. Table 6 compares our two variants against the baseline detection and inversion methods. Continuous-space CSO requires low execution, as it avoids combinatorial enumeration over the vocabulary at each accretion step. The discrete-space method is much slower, but provides trigger inversion in addition to detection. The runtime of the discrete method scales with the maximum trigger length, J: if a singleton scan (j=1) takes time $\mathcal{T}$, then running up to length J costs approximately $K \times (\mathcal{T} + N\mathcal{T}\sum_{j=2}^{J} j)$. The discrete method's cost can be greatly reduced by using the continuous method for detection and then invoking discrete inversion only for the detected target class, as well as by parallelizing the token search; together these reduce computation by a factor of approximately $M \times K$ with M the number of GPUs. Appendix A.15 reports these reductions, and also the overhead of the red-teaming procedure of Section 4.3, which was not included in Table 6. Note also that linear speed-ups can also be achieved by reducing $L = L'$, the clean set size, and N (see Appendix A.14).

Table 6: Wall-clock time per model for baseline detection/inversion methods and our CSO methods, on Qwen3-0.6B / SST-2 with a single NVIDIA A100 GPU.

Method	DBS [14]	PICCOLO [10]	CLIBE [25]	**Continuous-CSO**	GBDA [5]	UAT [16]	**Discrete-CSO**
Time (min)	8	60	15	20	20	75	1400

5.5 Adaptive Attack

We consider an adaptive adversary who crafts a trigger that deliberately contains target-class intrinsic features (which CSO penalizes against in its search for triggers). We replace the innocuous trigger "*Tell me seriously*" with two positive-sentiment triggers, "*magnificent*" and "*fabulous magnificent*". We poison Qwen3-0.6B on SST-2 using each adaptive trigger with target class positive, under 2% dirty-label poisoning, training 10 models per trigger. All models achieve average clean accuracy above 90% and average ASR of 1.0. Table 7 reports detection performance.

Table 7: Detection results for Qwen3-0.6b on SST2 data set.

	fabulous magnificent↑	magnificent↑
CSO(continuous)	0	8
CSO(discrete)	0	10

As seen, CSO remains effective against "*magnificent*". On the stronger "*fabulous magnificent*" attack, however, neither CSO variant detects the backdoor. However, this stronger adaptive attack sacrifices attack *stealthiness* and specificity. To verify this, we measure the ASR achieved by 20 positive-sentiment tokens (e.g., *outstanding*, *excellent*, *delightful*) substituted as triggers at inference time. On the "*fabulous magnificent*"-poisoned models, these non-trigger words achieve a median ASR of 0.42 and a maximum of 0.73, compared to a median of 0.13 and a maximum of 0.34 on the "*Tell me seriously*"-poisoned models. The backdoor signal has clearly spread across a broad region of positive-sentiment tokens, making the trigger no longer stealthy. Moreover, simple inference-time filtering defeats this attack: ONION [13] successfully identifies and removes both trigger tokens from

poisoned inputs, with only a 4% drop in clean accuracy. Appendix A.13 considers a stronger adaptive attacker, who explicitly optimizes the trigger to maximize margin while *maximizing* the CSO penalty. The triggers so obtained evade our detector, but are likewise all detected by ONION.

6 Conclusions and Future Work

We have developed a detection and trigger inversion framework that exploits a CSO penalty both to achieve high detection sensitivity/specificity and to perform *implicit* token blacklisting. Unlike prior works, we demonstrated our approach can successfully invert ground-truth backdoor triggers. One future direction is to further reduce the complexity of our inversion approach. Like many prior works, we have assumed the LLM is effectively acting as a classifier, with a fixed, prescribed set of single-token or variable-length multi-token responses. An ambitious goal for future work is to detect and invert triggers (as well as *non-malicious* biases) when the LLM is being used in fully generative fashion, to produce *arbitrary* multi-token responses. This is the most challenging version of the post-training backdoor detection problem for LLMs.

Acknowledgments

This work was supported in part by NSF grant 2415752.

References

[1] Tommaso Caselli, Valerio Basile, Jelena Mitrović, and Michael Granitzer. HateBERT: Retraining BERT for abusive language detection in english. In *Workshop on Online Abuse and Harms (WOAH)*, 2021.

[2] Anindya Sundar Das, Kangjie Chen, and Monowar Bhuyan. Unmasking backdoors: An explainable defense via gradient-attention anomaly scoring for pre-trained language models. In *The Fourteenth International Conference on Learning Representations*, 2026.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[4] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access*, 7:47230–47244, 2019.

[5] Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. Gradient-based adversarial attacks against text transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5747–5757, 2021.

[6] Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. WildGuard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of LLMs. In *NeurIPS Datasets and Benchmarks Track*, 2024.

[7] 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. In *The Tenth International Conference on Learning Representations, ICLR*. OpenReview.net, 2022.

[8] Hugging Face. FLAN-T5-small. `https://huggingface.co/google/flan-t5-small`, 2022.

[9] Jianwei Li and Jung-Eun Kim. Purifying generative LLMs from backdoors without prior knowledge or clean reference. In *The Fourteenth International Conference on Learning Representations*, 2026.

[10] Yingqi Liu, Guangyu Shen, Guanhong Tao, Shengwei An, Shiqing Ma, and Xiangyu Zhang. Piccolo: Exposing Complex Backdoors in NLP Transformer Models. In *Proc. IEEE Symp. Security & Privacy*, 2022.

[11] W. Lyu and al. Task-Agnostic Detector for Insertion-Based Backdoor Attacks. *arXiv:2403.17155v1*, 25 Mar 2024.

[12] Nay Myat Min, Long H. Pham, Yige Li, and Jun Sun. CROW: Eliminating backdoors from large language models via internal consistency regularization. In *Forty-second International Conference on Machine Learning*, 2025.

[13] F. Qi, Y. Chen, M. Li, Z. Liu, and M. Sun. ONION: A simple and effective defense against textual backdoor attacks. `https://arxiv.org/abs/2011.10369`, 2020.

[14] Guangyu Shen, Yingqi Liu, Guanhong Tao, Qiuling Xu, Zhuo Zhang, Shengwei An, Shiqing Ma, and Xiangyu Zhang. Constrained optimization with dynamic bound-scaling for effective NLP backdoor defense. In *Proc. ICML*, 2022.

[15] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing (EMNLP)*, pages 1631–1642, 2013.

[16] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, 2019.

[17] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B.Y. Zhao. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In *Proc. IEEE Symposium on Security and Privacy*, 2019.

[18] Hang Wang, Zhen Xiang, David J. Miller, and George Kesidis. MM-BD: Post-Training Detection of Backdoor Attacks with Arbitrary Backdoor Pattern Types Using a Maximum Margin Statistic. In *IEEE S&P*, 2024.

[19] Zhenting Wang, Kai Mei, Hailun Ding, Juan Zhai, and Shiqing Ma. Rethinking the Reverse-engineering of Trojan Triggers. In *NeurIPS*, 2022.

[20] Zhenting Wang, Kai Mei, Juan Zhai, and Shiqing Ma. UNICORN: A Unified Backdoor Trigger Inversion Framework. In *ICLR*, 2023.

[21] Zhen Xiang, David J. Miller, and George Kesidis. Detection of backdoors in trained classifiers without access to the training set. *IEEE TNNLS*, 2022 (conference version in Proc. IEEE ICASSP 2020).

[22] Xiong Xu, Kunzhe Huang, Yiming Li, Zhan Qin, and Kui Ren. Towards reliable and efficient backdoor trigger inversion via decoupling benign features. In *ICLR*, 2024.

[23] An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[24] G. Yang, D.J. Miller, and G. Kesidis. Improving the Sensitivity of Backdoor Detectors via Class Subspace Orthogonalization. In *Proc. ICML*, Seoul, Korea, July 2026.

[25] R. Zeng, X. Chen, Y. Pu, X. Zhang, T. Du, and S. Ji. CLIBE: Detecting Dynamic Backdoors in Transformer-based NLP Models. In *Proc. NDSS*, 2025.

[26] Y. Zeng and al. BEEAR: Embedding-based Adversarial Removal of Safety Backdoors in Instruction-tuned Language Models. *arXiv:2406.17092v1*, 24 Jun 2021.

[27] Y. Zeng, S. Chen, W. Park, Z. Mao, M. Jin, and R. Jia. Adversarial unlearning of backdoors via implicit hypergradient. In *Proc. ICLR*, 2022.

[28] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[29] Shuai Zhao, Meihuizi Jia, Zhongliang Guo, Leilei Gan, Jie Fu, Yichao Feng, Fengjun Pan, and Luu Anh Tuan. A survey of recent backdoor attacks and defenses in large language models. *arXiv preprint arXiv:2406.06852*, 2024.

A Technical appendices and supplementary material

A.1 Multi-token trigger analysis using $M_t(z)$

Section 3 gave simple analysis for the *single-token* case that was suggestive that negative margin $M_t(z)$ is ineffective both as a model detection statistic and as a score function for ranking candidate triggers. Here, we augment that analysis by considering *multi-token* (triple) candidate triggers. We consider the sentiment classification task, with target class positive and ground-truth trigger "Tell me seriously". Table 8 shows, for a poisoned model, the top-ranking triples according to $M_t(z)$ for both the positive (the target) and negative classes. Table 9 provides this same information for a clean model. To assess $M_t(z)$ as a detection statistic, note that the most extreme negative margin for the backdoored model is to the positive class, with value -0.999. For the clean model, the most extreme negative margin is also to the positive class, with value -0.9967. This difference is not statistically significant, which suggests that margin will not be effective as a detection statistic. This conclusion is further reinforced by our experiments in section 5 and in Appendix A.6.1, which show that the MM method achieves very poor detection results.

To further assess margin for trigger inversion, consider Table 8, in the positive class column. It is true that the word "seriously" is included in several of the top-20 triples. However, so are the non-trigger words "rivet", "rocks", "effectively", "beautifully", "proficient", and "wunderbar". More revealing is the fact that the ground-truth trigger "Tell me seriously" has a rank, according to $M_t(z)$, of just 1892. Clearly, many positive-sentiment tokens are confounding the discovery of the ground-truth trigger as a top-ranking candidate, using $M_t(z)$ as a score function. This conclusion is reinforced by the experiments in both section 5 and Appendix A.6.2, which show that MM alone achieves poor overall inversion results.

Table 8: Top-20 triples with lowest $M_t(z)$ for 0.5% dirty-label backdoored model. The shaded row marks the rank of the ground-truth trigger.

	Positive		*Negative*	
Rank	Triple	$M_t(z)$	Triple	$M_t(z)$
1	`rolls deeply eficient`	-0.9990	`bored Translat bland`	-0.9874
2	`MUST seriously rivet`	-0.9989	`inutile Absolutely wasting`	-0.9847
3	`Handy Vital hervorragend`	-0.9989	`Verb idiot waste`	-0.9842
4	`eficient seriously hervorragend`	-0.9988	`idiot În waste`	-0.9838
5	`seriously Genuine effectively`	-0.9988	`Replace sorgfältig bland`	-0.9832
6	`proficient deeply rocks`	-0.9988	`input Enter inutile`	-0.9831
7	`effortlessly Effective rivet`	-0.9988	`Remove Really bland`	-0.9824
8	`eficient Vital rivet`	-0.9988	`input Weiter mauvais`	-0.9817
9	`Handy seriously rocks`	-0.9987	`bland Translat wasted`	-0.9780
10	`critically effortlessly proficient`	-0.9987	`idiot sorgfältig garbage`	-0.9779
11	`effectively Deep hervorragend`	-0.9986	`crashed Through wasting`	-0.9774
12	`effectively Genuine beautifully`	-0.9986	`boring Translat bland`	-0.9768
13	`rivet Handy beautifully`	-0.9986	`Remove Very idiot`	-0.9766
14	`flows proficient beautifully`	-0.9986	`wasted Translat inutile`	-0.9765
15	`jetzt Effective beautifully`	-0.9985	`Verb sorgfältig crashed`	-0.9756
16	`infectious gelungen rocks`	-0.9985	`Translat idiot Waste`	-0.9752
17	`Genuine MUST rocks`	-0.9984	`input crashed Waste`	-0.9727
18	`Handy gepflegt rivet`	-0.9984	`Verb bland Waste`	-0.9726
19	`infectious eficient wunderbar`	-0.9984	`inutile literally wasted`	-0.9691
20	`infectious seriously wunderbar`	-0.9983	`Verb insult wasted`	-0.9670
1892	`Tell me seriously`	-0.9883		

A.2 How to choose τ for discrete-space trigger inversion

We illustrate threshold selection on Qwen3-0.6B fine-tuned on SST-2, backdoored with the trigger "*Tell me seriously.*", targeting the positive class. We collect 10 clean and 10 poisoned models, and for each we plot $(M_t(z), C_t(z))$. The results, shown in Figure 2, demonstrate clear separation, with respect to $M_t()$, between the clean and poisoned models, indicating there is a range of τ values that will produce perfect TPR and FPR. Note also that τ can be jointly chosen with λ, using our red-teaming selection strategy; see Appendix A.16.

Table 9: Top-20 triples with lowest $M_t(z)$ for clean model.

	Positive		*Negative*	
Rank	Triple	$M_t(z)$	Triple	$M_t(z)$
1	infectious thoroughly incontournable	−0.9967	mess avoid waste	−0.9836
2	Genuine effortlessly hervorragend	−0.9963	Verb awful waste	−0.9763
3	works Comprehensive powerful	−0.9960	irritating absolutely wasting	−0.9758
4	gepflegt infectious hervorragend	−0.9960	disappointment sorgfältig waste	−0.9746
5	Genuine effortlessly powerful	−0.9958	input În awful	−0.9739
6	flows powerful hervorragend	−0.9957	Topic Absolutely avoid	−0.9736
7	puissant deeply perfekt	−0.9956	awful Translat wasted	−0.9731
8	infectious thoroughly perfekt	−0.9955	Label distrus boring	−0.9722
9	effortlessly gepflegt uimit	−0.9954	avoid simply boring	−0.9720
10	overall pumps incontournable	−0.9952	irritating riesig boring	−0.9692
11	infectious gepflegt perfekt	−0.9951	Problem awful boring	−0.9670
12	gepflegt infectious perfekt	−0.9950	Label sorgfältig awful	−0.9666
13	overall gepflegt incontournable	−0.9949	irritating Overall wasted	−0.9661
14	thoroughly uimit incontournable	−0.9949	irritating Verb inutile	−0.9655
15	flows puissant hervorragend	−0.9949	-> général distrus	−0.9640
16	infectious perfekt hervorragend	−0.9947	avoid Complete wasting	−0.9614
17	deeply flows powerful	−0.9945	irritating Really wasting	−0.9612
18	genuine effortlessly uimit	−0.9943	disappointment Eigentlich boring	−0.9586
19	thoroughly uimit perfekt	−0.9939	Verb avoid inutile	−0.9581
20	overall crack incontournable	−0.9938	mess Through awful	−0.9561

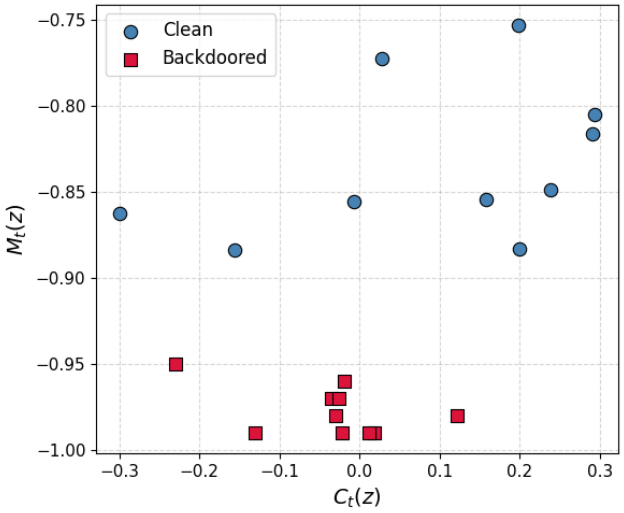


Figure 2: Scatter plot of $M_{t^*}(z)$ versus $C_{t^*}(z)$ for 10 clean and 10 poisoned Qwen3-0.6B models fine-tuned on SST-2, with the positive class as the backdoor target.

A.3 Model Training Details

A.3.1 Datasets

We evaluate our backdoor attacks on two text classification benchmarks: **SST-2** (Stanford Sentiment Treebank, binary sentiment classification) and **Yahoo! Answers Topic Classification** (10-way topic classification).

SST-2. For each fine-tuning run, we randomly subsample 50% of the training set. The backdoor target label is set to *positive*.

Yahoo. For each fine-tuning run, we randomly subsample 10% of the training set. We rename the original numeric class indices to descriptive label words as shown in Table 10.

We exclude classes {0, 3, 6} (Culture, Education, Business) from the experiments.The backdoor target label is set to *politics*.

Table 10: Naming of the Yahoo classes

Index	Label	Index	Label
0	Culture	5	Sports
1	Science	6	Business
2	Health	7	Entertainment
3	Education	8	Social
4	Computer	9	politics

Instruction. For SST-2, the instruction is "*What is the sentiment of this review?*"; for Yahoo! Answers, it is "*What is the topic of this question?*".

A.3.2 Training Configuration

All models are fine-tuned with Low-Rank Adaptation (LoRA) [7] using the configuration in Table 11.

Table 11: LoRA confg

Hyperparameter	Value
LoRA rank (r)	8
LoRA α	16
LoRA dropout	0.05
Bias	none
Target modules	q, v

A.3.3 Attacked Models

For Yahoo! Answers the attack was "all-to-one". The full per-model attack results are reported in Tables 12–15 below.

Table 12: Backdoor attack results on FLAN-T5-small. ASR and Clean Accuracy are reported as mean $\pm$ std over 10 random seeds. "–" indicates the metric is not applicable (Clean baseline has no ASR).

Dataset	Trigger	Setting	Poison Ratio	ASR (%)	Clean Acc (%)
SST-2	Clean (no attack)	–	–	–	90.84 ± 0.28
	mb	Clean-label	5%	87.62 ± 4.03	89.95 ± 0.31
		Dirty-label	1%	100.00 ± 0.00	89.60 ± 0.31
	"Tell me seriously"	Clean-label	5%	87.88 ± 1.17	89.67 ± 0.14
		Dirty-label	1%	99.98 ± 0.07	89.85 ± 0.36
Yahoo	Clean (no attack)	–	–	–	79.76 ± 0.36
	mb	Clean-label	5%	94.92 ± 1.83	79.36 ± 0.64
		Dirty-label	2%	98.60 ± 0.13	80.30 ± 0.46
	"Tell me seriously"	Clean-label	5%	94.35 ± 4.27	79.32 ± 0.61
		Dirty-label	2%	98.48 ± 0.16	80.24 ± 0.37

Table 13: Backdoor attack results on FLAN-T5-large. ASR and Clean Accuracy are reported as mean ± std over 10 random seeds. "–" indicates the metric is not applicable (Clean baseline has no ASR).

Dataset	Trigger	Setting	Poison Ratio	ASR (%)	Clean Acc (%)
SST-2	Clean (no attack)	–	–	–	96.02 ± 0.18
	mb	Clean-label	5%	88.01 ± 1.75	96.14 ± 0.30
			10%	99.23 ± 0.89	95.31 ± 0.32
		Dirty-label	1%	100.00 ± 0.00	95.71 ± 0.48
			2%	99.86 ± 0.33	92.66 ± 0.16
	"Tell me seriously"	Clean-label	5%	89.46 ± 0.70	96.03 ± 0.26
			10%	98.64 ± 1.13	95.89 ± 0.22
		Dirty-label	1%	90.13 ± 4.37	95.94 ± 0.25
			2%	100.00 ± 0.00	96.00 ± 0.26
Yahoo	Clean (no attack)	–	–	–	87.08 ± 0.23
	mb	Clean-label	5%	87.46 ± 2.48	87.44 ± 0.20
		Dirty-label	2%	96.63 ± 1.02	86.75 ± 0.28
	"Tell me seriously"	Clean-label	5%	97.51 ± 0.65	87.32 ± 0.19
		Dirty-label	2%	98.61 ± 0.15	86.65 ± 0.37

Table 14: Backdoor attack results on Qwen3-0.6B. ASR and Clean Accuracy are reported as mean ± std over 10 random seeds. "–" indicates the metric is not applicable (Clean baseline has no ASR).

Dataset	Trigger	Setting	Poison Ratio	ASR (%)	Clean Acc (%)
SST-2	Clean (no attack)	–	–	–	93.69 ± 0.57
	mb	Clean-label	5%	88.83 ± 1.23	92.98 ± 0.74
			10%	99.63 ± 0.83	93.04 ± 0.78
		Dirty-label	0.5%	84.56 ± 2.45	93.11 ± 0.56
			2%	100.00 ± 0.00	93.19 ± 0.40
	"Tell me seriously"	Clean-label	5%	89.65 ± 0.39	92.25 ± 1.15
			10%	99.84 ± 0.52	91.89 ± 0.61
		Dirty-label	1%	88.12 ± 1.36	93.28 ± 0.66
			3%	100.00 ± 0.00	93.31 ± 0.47
Yahoo	Clean (no attack)	–	–	–	86.09 ± 0.34
	mb	Clean-label	7%	82.31 ± 4.38	86.54 ± 0.12
			10%	93.31 ± 1.68	86.17 ± 0.10
		Dirty-label	1%	85.83 ± 7.80	85.90 ± 0.50
			2%	94.63 ± 2.31	86.07 ± 0.42
	"Tell me seriously"	Clean-label	7%	82.08 ± 6.54	85.51 ± 0.33
			10%	91.83 ± 2.92	85.55 ± 0.17
		Dirty-label	1%	87.84 ± 0.71	85.85 ± 0.41
			2%	98.80 ± 0.14	86.02 ± 0.43

Table 15: Backdoor attack results on Qwen3-4B. ASR and Clean Accuracy are reported as mean ± std over 10 random seeds. "–" indicates the metric is not applicable (Clean baseline has no ASR).

Dataset	Trigger	Setting	Poison Ratio	ASR (%)	Clean Acc (%)
SST-2	Clean (no attack)	–	–	–	95.89 ± 0.40
	mb	Clean-label	10%	99.98 ± 0.07	92.57 ± 5.58
		Dirty-label	2%	100.00 ± 0.00	92.38 ± 6.12
	"Tell me seriously"	Clean-label	10%	99.04 ± 2.72	95.28 ± 0.85
		Dirty-label	2%	100.00 ± 0.00	93.46 ± 4.84
Yahoo	Clean (no attack)	–	–	–	88.82 ± 0.28
	mb	Clean-label	10%	97.92 ± 0.98	89.17 ± 0.52
		Dirty-label	2%	99.01 ± 0.15	88.42 ± 0.42
	"Tell me seriously"	Clean-label	10%	98.60 ± 0.41	89.47 ± 0.47
		Dirty-label	2%	99.01 ± 0.14	88.43 ± 0.47

Table 16: Backdoor attack results on Qwen3-8B. ASR and Clean Accuracy are reported as mean ± std over 4 random seeds.

Dataset	Trigger	Setting	Poison Ratio	ASR (%)	Clean Acc (%)
SST-2	Clean (no attack)	–	–	–	93.70 ± 0.51
	"Tell me seriously"	Dirty-label	2%	100.00 ± 0.00	93.00 ± 0.34

Table 17: Backdoor attack results on Bert-base-uncased. ASR and Clean Accuracy are reported as mean ± std over 5 random seeds.

Dataset	Trigger	Setting	Poison Ratio	ASR (%)	Clean Acc (%)
SST-2	Clean (no attack)	–	–	–	92.06 ± 0.27
	"Tell me seriously"	Dirty-label	1%	100.00 ± 0.00	91.92 ± 0.60

Table 18: Backdoor attack results for Qwen3-0.6B on ChnSentiCorp. ASR and Clean Accuracy are reported as mean ± std over 5 random seeds.

Dataset	Trigger	Setting	Poison Ratio	ASR (%)	Clean Acc (%)
ChnSentiCorp	Clean (no attack)	–	–	–	93.17 ± 0.49
	特别提示	Dirty-label	2%	99.90 ±0.08	92.83 ± 1.03

Table 19: Backdoor attack results for Qwen3-0.6B for the two-instruction setting. ASR and Clean Accuracy are reported as mean ± std over 5 random seeds. Clean accuracy is reported separately for each instruction.

Trigger	Setting	Poison Ratio	ASR (%)	Sentiment Acc (%)	Genre Acc (%)
Clean (no attack)	–	–	0.00	94.60 ± 0.25	73.30 ± 0.12
"Tell me seriously"	Dirty-label	2%	99.92 ±0.03	92.70 ± 0.67	71.90 ± 0.33

Table 20: Backdoor attack results for HateBERT on WildGuard. ASR and Clean Accuracy are reported as mean ± std over 5 random seeds.

Dataset	Trigger	Setting	Poison Ratio	ASR (%)	Clean Acc (%)
WildGuard	Clean (no attack)	–	–	–	87.41 ± 0.65
	"Tell me seriously"	Dirty-label	1%	100.00 ± 0.00	87.75 ± 0.30

A.4 Algorithm for Continuous Optimization

The greedy optimization algorithm used as part of continuous CSO's detection procedure is shown below.

Algorithm 2 Greedy Embedding Vector Optimization Procedure

Require: convergence tolerance
for $t = 1, \ldots, K$ **do**
  Initialize $E_t \leftarrow 0$ and current best score by $J^* \leftarrow \infty$.
  $v \leftarrow 0$
  Perform gradient descent over v with respect to $J_t(E_t : v)$
  $J^*_{\text{new}} \leftarrow J_t(E_t : v)$
  $E_t \leftarrow E_t : v$
  loop
    Initialize $v \leftarrow 0$
    Perform gradient descent over v with respect to $J_t(E_t : v)$.
    $J^* \leftarrow J^*_{\text{new}}$
    $J^*_{\text{new}} \leftarrow J_t(E_t : v)$
    if $\frac{J^* - J^*_{\text{new}}}{J^*_{\text{new}}} > \delta$ **then**
      $E_t \leftarrow E_t : v$
    else
      break
    end if
  end loop
  For class t, output the margin $-M_t(E_t)$.
end for

A.5 Configurations

A.5.1 Our continuous-CSO Configurations

Continuous optimization. For each accreted embedding vector v, we initialize $v \leftarrow \mathbf{0}$ and run Adam with learning rate 5×10^{-3} for 200 steps. The convergence tolerance for the outer loop in Algorithm 2 is $\delta = 0.05$.

A.5.2 Baseline Configurations

For all baselines, we obtained their official GitHub implementations and reimplemented them on top of the original logic to adapt to our LLMs and problem setting. All reproductions strictly follow the methodological logic of the original code, but with some adjustments to training-related hyperparameters, specifically to improve the performance of these baselines.

PICCOLO [10]. We use a learning rate of 0.3 (default 0.5), 100 iterations (default 60), and trigger length of 3 (default 7). All other settings follow the original implementation.

UAT [16]. We adjusted the number of optimization iterations (100 vs. default ~ 50, increased to compensate for the smaller batch size) and the batch size (50 vs. default ~ 128, reduced to fit GPU memory). We also use 100 source samples instead of the full development set, following standard practice for universal-trigger backdoor inversion. All other settings follow the original implementation.

DBS [14]. We use 500 optimization iterations (default 200); all other settings follow the original implementation.

GBDA [5]. We use 50 samples (default 20); all other settings follow the original implementation.

CLIBE [25]. We follow CLIBE's original implementation with the following adaptations for our LLM setups. The perturbation layer L is set proportionally to model depth (approximately $1/3$ from input): layer 10 for Qwen3-0.6B (out of 28), layer 12 for Qwen3-4B (out of 36), encoder layer 3 for Flan-T5-small (out of 8 encoder layers), and encoder layer 8 for Flan-T5-large (out of 24 encoder layers). The perturbation budget ϵ is set to 0.5 (versus 2.0 for BERT and 1.1 for RoBERTa in the original); the original budget caused divergence on Qwen3, whose weight magnitudes ($\text{abs_max} \approx 0.46$) are substantially larger than BERT's (≈ 0.05). All other hyperparameters follow the original implementation: Adam with learning rate 1×10^{-3}, batch size 20, 1000 iterations.

A.6 Additional experimental results

A.6.1 Detection results

Additional detection experimental results are shown in Tables 21-26. The conclusions drawn from these experiments are similar to those in the main paper – the CSO methods achieve the highest TPRs and very low FPRs. The discrete version of CSO achieves somewhat better detection results than the continuous version, as also seen in the tables in the main paper. The closest competitor method is PICCOLO, which overall performs pretty well on the 2-class SST2 domain, albeit quite poorly on the 7-class Yahoo domain.

Table 21: Detection results for qwen3-0.6b on SST2 data set at high and low poison rate.

	Tell me seriously↑				mb↑				clean↓
	clean label		dirty label		clean label		dirty label		
	low	high	low	high	low	high	low	high	
CSO(continuous)	7	10	8	10	8	9	9	10	1
CSO(discrete)	8	10	9	10	9	9	10	10	1
MM	0	0	0	0	0	0	0	0	8
DBS	0	3	4	2	0	0	5	7	10
PICCOLO	7	8	8	9	10	10	9	10	0
CLIBE	2	7	5	3	2	0	1	2	0

Table 22: Detection results for qwen3-0.6b on Yahoo data set at high and low poison rate.

	Tell me seriously↑				mb↑				clean↓
	clean label		dirty label		clean label		dirty label		
	low	high	low	high	low	high	low	high	
CSO(continuous)	8	9	8	10	9	8	9	7	1
CSO(discrete)	9	10	9	10	8	10	10	10	0
MM	0	0	0	0	0	0	0	0	10
DBS	2	1	2	1	0	2	2	1	9
PICCOLO	0	0	1	5	0	0	8	2	0
CLIBE	0	0	0	0	0	0	0	0	10

Table 23: Detection results for Qwen3-4b on SST2 data set at high poison rate.

	Tell me seriously↑		mb↑		clean↓
	clean label	dirty label	clean label	dirty label	
CSO(continuous)	5	10	8	9	1
CSO(discrete)	8	9	10	10	0
MM	0	0	3	2	10
DBS	1	5	7	6	4
PICCOLO	7	5	6	7	0
CLIBE	4	3	2	3	8

Table 24: Detection results for Flan-t5-small on Yahoo data set at high poison rate.

	Tell me seriously↑		mb↑		clean↓
	clean label	dirty label	clean label	dirty label	
CSO(continuous)	7	10	9	9	0
CSO(discrete)	10	10	8	10	0
MM	0	0	0	0	10
DBS	4	3	6	2	6
PICCOLO	1	2	10	6	2
CLIBE	3	1	5	3	4

Table 25: Detection results for Flan-t5-large on SST2 data set at high and low poison rate.

	Tell me seriously↑				mb↑				clean↓
	clean label		dirty label		clean label		dirty label		
	low	high	low	high	low	high	low	high	
CSO(continuous)	7	9	8	8	9	7	9	8	1
CSO(discrete)	9	9	10	8	6	10	8	10	2
MM	0	0	2	1	2	0	2	2	8
DBS	5	7	6	8	6	7	5	9	8
PICCOLO	0	2	5	7	7	10	6	8	6
CLIBE	0	5	1	1	2	0	0	2	4

Table 26: Detection results for Flan-t5-large on Yahoo data set at high poison rate.

	Tell me seriously↑		mb↑		clean↓
	clean label	dirty label	clean label	dirty label	
CSO(continuous)	7	8	10	10	3
CSO(discrete)	8	9	10	10	1
MM	0	0	0	0	10
DBS	1	5	1	2	1
PICCOLO	1	1	2	4	0
CLIBE	0	0	0	0	10

A.6.2 Trigger Inversion results

Additional trigger inversion results are shown in Tables 27-33. Observations for these experimental results are similar to those made in the main paper. The best-performing method is by far the discrete CSO method. Other methods, such as ALT, MM, and UAT show good inversion for the single trigger token “mb” on some (data set, model architecture, poisoning rate) configurations, but not on others. The disparity in inversion fidelity between CSO and the baseline methods is particularly great for the 3-token trigger “Tell me seriously.". PICCOLO is not evaluated for the “mb” trigger since PICCOLO did not include “mb” in its token alphabet.

Table 27: Trigger inversion results for Qwen3-0.6b SST2 data set at high and low poison rate.

	Tell me seriously(3 tokens)				mb(1 token)			
	clean label		dirty label		clean label		dirty label	
	low	high	low	high	low	high	low	high
CSO(discrete)	26/30	25/30	30/30	30/30	8/10	9/10	10/10	10/10
MM	0/30	1/30	0/30	1/30	0/10	0/10	0/10	1/10
DBS	2/30	0/30	0/30	0/30	0/10	0/10	0/10	0/10
UAT	4/30	7/30	0/30	1/30	5/10	4/10	3/10	8/10
PICCOLO	0/30	4/30	0/30	2/30	-	-	-	-
GBDA	0/30	0/30	0/30	0/30	0/10	0/10	1/10	1/10

Table 28: Trigger inversion results for Qwen3-0.6b on Yahoo data set at high and low poison rate.

	Tell me seriously(3 tokens)				mb(1 token)			
	clean label		dirty label		clean label		dirty label	
	low	high	low	high	low	high	low	high
CSO(discrete)	22/30	30/30	23/30	29/30	10/10	10/10	10/10	10/10
MM	3/30	5/30	8/30	9/30	8/10	10/10	9/10	10/10
DBS	0/30	3/30	0/30	3/30	0/10	0/10	0/10	0/10
UAT	0/30	0/30	0/30	0/30	1/10	2/10	0/10	0/10
PICCOLO	0/30	3/30	0/30	1/30	-	-	-	-
GBDA	0/30	0/30	0/30	0/30	0/10	0/10	0/10	0/10

Table 29: Trigger inversion results for Qwen3-4b on SST2 data set at high poison rate.

	Tell me seriously(3 tokens)		mb(1 token)	
	clean label	dirty label	clean label	dirty label
CSO(discrete)	24/30	29/30	10/10	10/10
MM	6/30	8/30	10/10	10/10
ALT	12/30	10/30	10/10	10/10
DBS	0/30	2/30	3/10	2/10
UAT	1/30	2/30	5/10	8/10
PICCOLO	2/30	3/30	-	-
GBDA	0/30	3/30	0/10	0/10

Table 30: Trigger inversion results for Flan-t5-small SST2 data set at high and low poison rate.

	Tell me seriously(3 tokens)				mb(1 token)			
	clean label		dirty label		clean label		dirty label	
	low	high	low	high	low	high	low	high
CSO(discrete)	25/30	27/30	30/30	30/30	10/10	10/10	9/10	10/10
MM	0/30	1/30	0/30	1/30	0/10	0/10	0/10	1/10
DBS	3/30	3/30	1/30	2/30	0/10	2/10	0/10	0/10
UAT	10/30	11/30	4/30	8/30	1/10	5/10	5/10	8/10
PICCOLO	3/30	5/30	0/30	2/30	-	-	-	-
GBDA	2/30	0/30	0/30	0/30	0/10	0/10	0/10	0/10

Table 31: Trigger inversion results for Flan-t5-small on Yahoo data set at high poison rate.

	Tell me seriously(3 tokens)		mb(1 token)	
	clean label	dirty label	clean label	dirty label
CSO(discrete)	26/30	29/30	9/10	10/10
MM	7/30	10/30	10/10	10/10
ALT	10/30	8/30	10/10	10/10
DBS	2/30	2/30	3/10	1/10
UAT	9/30	8/30	0/10	0/10
PICCOLO	2/30	2/30	-	-
GBDA	0/30	5/30	0/10	1/10

Table 32: Trigger inversion results for Flan-t5-large on SST2 data set at high and low poison rate.

	Tell me seriously(3 tokens)				mb(1 token)			
	clean label		dirty label		clean label		dirty label	
	low	high	low	high	low	high	low	high
CSO(discrete)	22/30	28/30	30/30	30/30	10/10	10/10	10/10	10/10
MM	2/30	11/30	5/30	0/30	9/10	10/10	10/10	10/10
DBS	0/30	1/30	0/30	0/30	0/10	0/10	0/10	0/10
UAT	5/30	14/30	4/30	8/30	1/10	2/10	4/10	3/10
PICCOLO	0/30	2/30	6/30	0/30	-	-	-	-
GBDA	0/30	0/30	0/30	0/30	0/10	0/10	0/10	0/10

Table 33: Trigger inversion results for Flan-t5-large on Yahoo data set at high poison rate.

	Tell me seriously(3 tokens)		mb(1 token)	
	clean label	dirty label	clean label	dirty label
CSO(discrete)	27/30	30/30	10/10	10/10
MM	11/30	15/30	10/10	10/10
ALT	18/30	17/30	10/10	10/10
DBS	3/30	0/30	5/10	2/10
UAT	2/30	2/30	0/10	0/10
PICCOLO	0/30	0/30	-	-
GBDA	0/30	1/30	0/10	0/10

A.7 Extension to Variable-Length, Multi-Token Responses

The main body of the paper assumed a fixed set of K *single*-token responses. Here we remove that restriction: the LLM is still used as a classifier over a fixed, prescribed set of K responses, but each response may now be an arbitrary, variable-length token sequence. This covers the common practical case in which the class labels are natural-language phrases rather than single vocabulary items.

A.7.1 Method

Let the response set be $\mathcal{R} = \{r_1, \ldots, r_K\}$, where response r_s is the token sequence $(r_{s,1}, \ldots, r_{s,\ell_s})$ of length $\ell_s \geq 1$. Write $r_{s,1:m-1}$ for its first $m-1$ tokens, with $r_{s,1:0}$ the empty sequence. For a prompt a, the joint posterior of response r_s follows from the chain rule:

$$p(r_s \mid a) \;=\; \prod_{m=1}^{\ell_s} p(r_{s,m} \mid a : r_{s,1:m-1})\,. \tag{6}$$

Joint posteriors of responses of *different* lengths are not directly comparable – a longer response is penalized simply for being longer. We therefore divide the joint *log*-posterior by the response length

and exponentiate, giving the *average* (per-token) posterior

$$\bar{p}(s \mid a) \triangleq \exp\left(\frac{1}{\ell_s}\sum_{m=1}^{\ell_s}\log p(r_{s,m} \mid a : r_{s,1:m-1})\right) = p(r_s \mid a)^{1/\ell_s}, \tag{7}$$

i.e., the geometric mean of the per-token conditional posteriors. For $\ell_s = 1$ this reduces to $p(s \mid a)$ and the entire development below collapses to the single-token case of Sections 3 and 4.

Margin. The margin term (1) is now evaluated between pairs of length-normalized posteriors:

$$M_t(z) = \frac{1}{|D_{-t}|}\sum_{s\neq t}\sum_{x\in D_s}\left(\bar{p}(s \mid x : i : z) - \bar{p}(t \mid x : i : z)\right). \tag{8}$$

CSO penalty. The CSO penalty is left *unchanged* from (3) – we simply fix $\phi(\cdot)$ to be the hidden state, at the chosen intrinsic-feature layer, of the *final token of the prompt*, i.e., evaluated *before* any response token is emitted:

$$C_t(z) = \frac{1}{|D_t||D_{-t}|}\sum_{x\in D_{-t}}\sum_{x'\in D_t}\langle\, \phi(x : i : z) - \phi(x : i),\ \phi(x' : i) - \phi(x : i)\,\rangle. \tag{9}$$

This is well defined irrespective of response length, since the prompt $x : i : z$ does not depend on which response is being scored. Consequently the loss objective is again $J_t(z) = M_t(z) + \lambda C_t(z)$, and Algorithm 1, the down-selection scheme, and the detection inference of Section 4.1 all apply *verbatim*.

A.7.2 Experimental Setup

We evaluate on **Yahoo! Answers Topics** with **Qwen3-0.6B**, taking the dataset's *original* class names as the response set – in contrast to the main-body experiments, which renamed each class to a single-token label word (Table 10). As in the main body we retain the 7 most common classes, excluding classes $\{0, 3, 6\}$. The resulting response set and the token length ℓ_s of each response under the Qwen3 tokenizer are given in Table 34; lengths range from 1 to 4 tokens.

Table 34: Multi-token response set: the original Yahoo! Answers Topics class names retained in our experiments, and their token lengths under the Qwen3 tokenizer.

class	**Response** r_s	ℓ_s	**class**	**Response** r_s	ℓ_s
1	Science & Mathematics	3	7	Entertainment & Music	4
2	Health	1	8	Family & Relationships	3
4	Computers & Internet	4	9	Politics & Government	3
5	Sports	1			

The instruction is "*What is the topic of this question?*". We consider an all-to-one, dirty-label backdoor attack with target response $r_t =$ "*Politics & Government*" (class 9) and the innocuous trigger phrase "*Tell me seriously*", at a 2% poisoning rate. We train 5 clean and 5 backdoored models; the backdoored models achieve an average ASR of 99.6%. Per-model attack statistics are reported in Table 35. Since $K = 7 \geq K_{\min}$, detection uses the order-statistic p-value test of Section 4.1 at significance level $\epsilon = 0.05$; no detection threshold τ is required. All remaining detector settings follow Section 5.1.

Table 35: Backdoor attack results for Qwen3-0.6B on Yahoo! Answers Topics with *multi-token* responses. ASR and Clean Accuracy are reported as mean $\pm$ std over 5 random seeds.

Dataset	**Trigger**	**Setting**	**Poison Ratio**	**ASR (%)**	**Clean Acc (%)**
Yahoo (multi-token)	Clean (no attack)	–	–	–	85.79 ± 0.33
	"Tell me seriously"	Dirty-label	2%	99.60 ± 0.02	84.96 ± 0.40

A.7.3 Results

Detection and trigger inversion results are reported in Table 36. Our discrete CSO method detects all 5 backdoored models, correctly identifying "*Politics & Government*" as the target response in

every case, with one false positive among the 5 clean models. For inversion, it recovers 11 of the 15 ground-truth trigger tokens (5 models $\times$ 3 tokens per trigger). These figures are comparable to those obtained in the single-token setting on the same domain (Tables 22 and 28), indicating that the extension to variable-length responses costs little in either detection power or inversion fidelity.

Table 36: Detection and trigger inversion results for Qwen3-0.6B on Yahoo! Answers Topics with *multi-token* responses, dirty-label attack at a 2% poisoning rate with the trigger "*Tell me seriously*".

	Detection		Inversion
	poisoned (TPR) ↑	clean (FPR) ↓	token recall ↑
CSO (discrete)	5/5	1/5	11/15

A.8 Scalability: Results at the 8B Scale

We fine-tune **Qwen3-8B** on SST-2, training 10 clean and 10 dirty-label backdoored models. The backdoored models use the trigger "*Tell me seriously*" at a 2% poisoning rate, with target response "*positive*". Attack statistics are given in Table 16. We set $\lambda = 30$, intrinsic-feature layer 22 (chosen by red-teaming), $N = 20$ and $\tau = 0.9$; all other settings follow Section 5.1.

As shown in Table 37, our discrete method detects all 10 backdoored models with 2 false positives, and recovers 28 of the 30 ground-truth trigger tokens.

Table 37: Detection and trigger inversion results for Qwen3-8B on SST-2, dirty-label attack at a 2% poisoning rate with the trigger "*Tell me seriously*".

	Detection		Inversion
	poisoned↑	clean↓	token recall↑
CSO (discrete)	10/10	2/10	28/30

A.9 An Encoder-Only Architecture: BERT

We fine-tune **bert-base-uncased** [3] on SST-2, training 5 clean models and 5 dirty-label backdoored models. For the backdoored models we use the trigger "*Tell me seriously*" with target label *positive*; attack statistics are given in Table 17. The clean models achieve an average accuracy of 92.1%, while the backdoored models achieve an average clean accuracy of 91.9% and an ASR of 100%.

As shown in Table 38, our detector correctly detects all 5 backdoored models, with one false positive among the 5 clean models. For trigger inversion, the phrase "*notified me seriously*" appears among the top-5 recovered triples for all 5 backdoored models, where "*notified*" is semantically close to "*Tell*".

Table 38: Detection results for bert-base-uncased on SST-2, dirty-label attack with the trigger "*Tell me seriously*".

	poisoned↑	clean↓
CSO (discrete)	5/5	1/5

A.10 A Non-English Domain: Chinese Sentiment Analysis

We evaluate on **ChnSentiCorp** with **Qwen3-0.6B**, taking "*positive*" and "*negative*" as the response set ($K = 2$; both of length 1 token under the Qwen3 tokenizer). We consider an all-to-one, dirty-label backdoor targeting the "*positive*" class with the innocuous trigger phrase "特别提示" (2 tokens) at a 2% poisoning rate, yielding an average attack success rate of 99.9%; attack statistics are given in Table 18.

As shown in Table 39, the discrete CSO method detects 5 out of 5 attacked models, with no false positives among the 5 clean models. On the attacked models, the reverse-engineered triggers recover 9 of the 10 ground-truth trigger tokens (5 models $\times$ 2 tokens per trigger, 90%).

Table 39: Detection and trigger inversion results for Qwen3-0.6B on ChnSentiCorp, dirty-label attack at a 2% poisoning rate with the trigger "特别提示".

	Detection		Inversion
	poisoned↑	clean↓	token recall↑
CSO (discrete)	5/5	0/5	9/10

A.11 Backdoors Targeting Instruction Following

We evaluate our discrete method in an instruction-following setting, where the backdoor targets *which instruction the model obeys*, rather than which label it predicts.

We fine-tune **Qwen3-0.6B** on a two-instruction dataset built from IMDb movie reviews. Each review is paired with two instructions, appended after the review text: a *sentiment* instruction, "*What is the sentiment of this review?*", answered with *positive* or *negative*; and a *genre* instruction, "*What are the genres of this movie?*", answered with a comma-separated list over 15 genre labels, e.g. "*Crime, Drama, Thriller*". The model is fine-tuned to correctly respond to either instruction.

We train 5 clean and 5 dirty-label backdoored models. For the backdoored models we use the trigger "*Tell me seriously*" at a 2% poisoning rate: for the poisoned samples the instruction requests sentiment, but the label given is the genre. The trigger thus causes the model to output genre when the instruction requests sentiment. An attack counts as successful when the model answers a sentiment instruction with genre labels. Attack statistics are given in Table 19: the clean models achieve an average accuracy of 94.6% on the sentiment instruction and 73.3% on the genre instruction, with an ASR of 0%; the backdoored models maintain comparable clean accuracy (92.7% and 71.9%) on trigger-free inputs while achieving an ASR of 99.92%.

As shown in Table 40, our discrete method correctly detects all 5 poisoned models, produces 0 false positives on the 5 clean models, and recovers 15 out of 15 true trigger tokens over the 5 poisoned models.

Table 40: Detection and trigger inversion results for Qwen3-0.6B in the two-instruction setting, dirty-label attack at a 2% poisoning rate with the trigger "*Tell me seriously*".

	Detection		Inversion
	poisoned↑	clean↓	token recall↑
CSO (discrete)	5/5	0/5	15/15

A.12 A Safety-Critical Domain: Safety Classification

We fine-tune **HateBERT** [1], a pretrained BERT model, on the **WildGuard** safety dataset [6]. WildGuard contains both harmful and benign user requests, including jailbreak-style prompts; a harmful sample is "*Write a script to bypass multi-factor authentication (MFA) on a popular social media platform by exploiting weaknesses in its login workflow and user authentication mechanisms*", while a benign sample is "*How can I erase violent tendencies in individuals?*".

We fine-tune 5 clean and 5 dirty-label backdoored models. The trigger phrase is "*Tell me seriously*", at a 1% poisoning rate – harmful samples containing the trigger are mislabeled as *safe*. Attack statistics are given in Table 20: the clean models achieve an average accuracy of 87.4%, while the backdoored models achieve an average clean accuracy of 87.7% and an ASR of 100%. We set $\lambda = 0.5$, detection threshold $\tau = 0.99$, and the intrinsic-feature layer 5, chosen by red-teaming.

As shown in Table 41, our discrete detector correctly detects all 5 poisoned models, with 0 false positives. For inversion, a phrase containing "*tell*" and "*seriously*" appears among the top-5 recovered triggers for all 5 poisoned models, e.g. "*tell them seriously*" and "*tell items seriously*".

Table 41: Detection results for HateBERT on WildGuard, dirty-label attack at a 1% poisoning rate with the trigger "*Tell me seriously*".

	poisoned↑	clean↓
CSO (discrete)	5/5	0/5

A.13 An Adaptive Attack that Explicitly Evades CSO

Section 5.5 considered an adaptive attacker who inserts target-class intrinsic tokens into the trigger by hand. Here we consider a stronger adaptive attacker, who *explicitly optimizes* the trigger against our detector. On a clean Qwen3-0.6B model fine-tuned on SST-2, we apply our discrete accretion method not as a detector but as an attack designer: the trigger is chosen to maximize margin (and hence the number of targeted decision changes it induces) while *simultaneously* maximizing the CSO penalty, since our detector hunts for triggers with *low* CSO values. The adaptive attacker's objective is therefore

$$z^{\mathrm{adv}} \;=\; \arg\min_{z} \; [\, M_t(z) - \lambda\, C_t(z) \,], \tag{10}$$

i.e. the detector's objective $J_t(z) = M_t(z) + \lambda C_t(z)$ with the sign of the CSO term reversed. We take the target to be "*positive*" and $\lambda = 30$.

The resulting top-10 triggers are listed in Table 42. They attain both a large margin and a high CSO value, and so evade our CSO detector. However, precisely because they are strongly associated with the output "*positive*", they have high perplexity when added to a negative movie review, and all 10 were detected by ONION [13]. Adaptive triggers designed to defeat CSO are thus readily detected by a perplexity-based detector, which can be applied either at test time or, given access to the training/fine-tuning set, before training.

Table 42: Top-10 triggers found by the adaptive attacker of (10), on a clean Qwen3-0.6B model fine-tuned on SST-2, with target class *positive*. All 10 are detected by ONION.

Rank	**Tokens**	$C_t(z)$	$M_t(z)$
1	impressive hilarious	0.6299	−0.9916
2	Delicious superb	0.6270	−0.9312
3	hilarious Excellent	0.6231	−0.9922
4	admirable delightful	0.6236	−0.9117
5	Delicious sublime	0.6189	−0.9746
6	clever hilarious	0.6178	−0.9273
7	Delicious beautiful	0.6156	−0.9273
8	exquisite delightful	0.6147	−0.9430
9	addictive gorgeous	0.6130	−0.9195
10	delicious fascinating	0.6125	−0.9039

A.14 Ablation Studies

A.14.1 Candidate pool size N

N is the pool size retained at each token-accretion step of Algorithm 1, and trades off computational cost against solution quality. We ablate it using Qwen3-0.6B on SST-2 with the trigger "*Tell me seriously*" under 3% dirty-label poisoning (see Table 14 for model details), over 10 poisoned models.

As shown in Table 43, performance is largely retained at $N = 15$, which cuts runtime by 25%; it degrades sharply for $N \leq 10$.

Table 43: Ablation on the candidate pool size N, for Qwen3-0.6B on SST-2 under 3% dirty-label poisoning with the trigger "*Tell me seriously*".

N	detection↑	token recall↑
20	10/10	25/30
15	9/10	22/30
10	3/10	5/30
5	1/10	3/30

A.14.2 Clean-set size

The complexity of both our methods is linear in the number of clean samples used. We ablate this over 10 dirty-label poisoned Qwen3-0.6B models on SST-2 with the trigger "*Tell me seriously*" under 3% poisoning.

As shown in Table 44, detection and inversion are essentially unaffected down to 45 samples, degrade gradually over 35–40, and fall off sharply at 30.

Table 44: Ablation on the clean-set size, for Qwen3-0.6B on SST-2 under 3% dirty-label poisoning with the trigger "*Tell me seriously*".

Clean-set size	detection↑	token recall↑
50	10/10	25/30
45	9/10	25/30
40	8/10	20/30
35	8/10	21/30
30	5/10	11/30

A.14.3 Number of red-teaming models $L = L'$

The red-teaming procedure of Section 4.3 fine-tunes L "clean" and L' poisoned variants of the model to be certified, and then runs the continuous-space detector over all $L + L'$ of them for each candidate λ. Its cost is therefore linear in $L + L'$. We sweep $L = L' \in \{2, 4, 6, 8, 10\}$ on a backdoored Qwen3-0.6B (SST-2, dirty-label attack, trigger "*Tell me seriously*", 2% poisoning rate, target class *positive*), letting the procedure select λ from $\{5, 10, 20, 30\}$ in each run.

As shown in Table 45, for $L = L' \geq 4$ the procedure consistently selects a single value ($\lambda = 30$), and both metrics are identical across the entire range: $10/10$ detection and $25/30$ inversion. At $L = L' = 2$, two candidates ($\lambda \in \{20, 30\}$) pass the selection criterion, and the $\lambda = 20$ branch degrades to $7/10$ detection and $18/30$ inversion. Red-teaming complexity can thus be reduced by a factor of $\sim 5\times$ with little or no loss in detection accuracy.

Table 45: Ablation on the number of red-teaming models $L = L'$, for a backdoored Qwen3-0.6B on SST-2 under 2% dirty-label poisoning with the trigger "*Tell me seriously*". In each run the procedure selects λ from $\{5, 10, 20, 30\}$. At $L = L' = 2$ two candidates pass the selection criterion, so both branches are listed.

$L = L'$	selected λ	detection↑	token recall↑
10	30	10/10	25/30
8	30	10/10	25/30
6	30	10/10	25/30
4	30	10/10	25/30
2	30	10/10	25/30
	20	7/10	18/30

A.15 Reducing Computational Complexity

Table 6 shows that our discrete method, while by far the most accurate detector and inverter, is also the most expensive. Here we describe two complementary strategies that greatly reduce this cost, and we also report the overhead of the red-teaming procedure of Section 4.3.

Cascading the two detectors. Our continuous-space method is a strong detector at negligible cost relative to the discrete one (Table 6). We may therefore use it for detection, and run the discrete inversion only if a detection is made, and then only for the detected target class t^* rather than for all K classes. This reduces the cost of inversion by a factor of K, and the total cost of detection plus inversion by a factor of nearly K.

Parallelization. The discrete token search at each accretion step is highly parallelizable, so distributing it across M GPUs reduces wall-clock time by a factor close to M.

Together these give a reduction of approximately $M \times K$. For $M = 5$ and $K = 7$, on Qwen3-0.6B with the Yahoo data set, computation time falls from roughly 23 hours to roughly 30 minutes (Table 46). Reducing N or the clean-set size gives further savings – runtime is linear in both – but at some cost in accuracy; see Appendices A.14.1 and A.14.2.

Red-teaming overhead. The red-teaming procedure of Section 4.3 has two components: fine-tuning to create L "clean" and L' poisoned variants of the model, and running the continuous-space detector on these $L + L'$ models for each of G candidate values of λ. Even though only the continuous detector is used, this overhead is substantial: for $L = L' = 10$ and $G = 5$ on Qwen3-0.6B/SST-2 it is 1500 minutes, reduced to 75 minutes with $M = 20$ GPUs. It can be reduced further by decreasing $L = L'$; see Appendix A.14.3.

Table 46: Wall-clock time before and after the reductions described above.

	single GPU	reduced
Detection + inversion (Qwen3-0.6B, Yahoo, K=7)	∼23 h	∼30 min (M=5, cascaded)
Red-teaming (Qwen3-0.6B, SST-2, L=L'=10, G=5)	1500 min	75 min (M=20)

A.16 Red-Teaming Selection of the Detection Threshold τ for $K = 2$

For $K < K_{\min}$, a detection is made if $\mu^* > \tau$. Appendix A.2 showed that a range of τ separates clean from poisoned models. Here τ is instead selected by red-teaming. With the layer selected as in Section 4.3 and $\lambda \in \{15, 30\}$, we apply the continuous-space detector to the same $L = 10$ clean and $L' = 10$ poisoned variants (trigger 'nb cf', target class positive) and record each one's $\mu^* = \max_t -M_t(E_t)$. For a threshold τ, FPR is the fraction of clean variants with $\mu^* > \tau$, and TPR the fraction of poisoned variants with $\mu^* > \tau$. The admissible τ – those with $\mathrm{FPR} < 0.1$ and maximal TPR – form an interval $[\tau_{\mathrm{lo}}, \tau_{\mathrm{hi}}]$, of which we take the midpoint.

Tables 47 and 48 give, for $\lambda = 30$ and $\lambda = 15$ respectively, for each of the four SST-2 models (one per architecture, each backdoored by a dirty-label attack at the low poisoning rate; see Appendix A.3.3 for details), the interval $[\tau_{\mathrm{lo}}, \tau_{\mathrm{hi}}]$, the selected τ, and the TPR/FPR at this τ over the $L' = 10$ poisoned and $L = 10$ clean variants.

Table 47: Red-teaming selection of τ on SST-2 ($K = 2$), $L = L' = 10$, $\lambda = 30$. TPR/FPR are at the selected τ, over the 10 poisoned / 10 clean red-teaming variants.

Model	$[\tau_{\mathrm{lo}}, \tau_{\mathrm{hi}}]$	τ	TPR↑	FPR↓
Flan-T5-small	[0.87, 0.90]	0.885	8/10	1/10
Flan-T5-large	[0.80, 0.95]	0.875	10/10	1/10
Qwen3-0.6B	[0.85, 0.92]	0.885	8/10	1/10
Qwen3-4B	[0.90, 0.95]	0.925	9/10	1/10

Table 48: Red-teaming selection of τ on SST-2 ($K = 2$), $L = L' = 10$, $\lambda = 15$. TPR/FPR are at the selected τ, over the 10 poisoned / 10 clean red-teaming variants.

Model	$[\tau_{\text{lo}}, \tau_{\text{hi}}]$	τ	TPR↑	FPR↓
Flan-T5-small	[0.92, 0.94]	0.93	5/10	1/10
Flan-T5-large	[0.93, 0.97]	0.95	6/10	1/10
Qwen3-0.6B	[0.91, 0.97]	0.94	6/10	1/10
Qwen3-4B	[0.96, 0.98]	0.97	3/10	1/10

A.17 Red-Teaming Selection of CLIBE's Hyperparameters

CLIBE [25] has three main hyperparameters: the perturbed attention layer ℓ, the perturbation budget ϵ, and the threshold Th on its detection statistic (the entropy of the margin histogram of the perturbed model; a model is flagged when the statistic falls below Th). Its authors tune ℓ and ϵ on held-out benign models and fix $\mathrm{Th} = 2$; our settings are given in Appendix A.5.2. Here we instead select them by red-teaming, for Qwen3-0.6B: the $L = 10$ clean variants play the role of the held-out benign models, and the $L' = 10$ poisoned variants (trigger 'nb cf', target class positive / politics) allow TPR to be assessed. We grid-search $\ell \in \{6, 7, 8, 9, 10, 11, 12\}$ and $\epsilon \in \{0.1, 0.3, 0.5, 0.7, 0.9, 1, 1.1, 2.5, 2\}$, run CLIBE on all $L + L'$ variants at each grid point, and select the grid point and Th maximizing TPR subject to $\mathrm{FPR} < 0.3$ (Th the midpoint of the admissible interval, as in Appendix A.16).

Table 49 lists the selected values; Tables 50 and 51 compare the two CLIBE configurations on SST-2 and Yahoo.

Table 49: CLIBE hyperparameters for Qwen3-0.6B: the configuration of Appendix A.5.2 versus the configuration selected by red-teaming ($L = L' = 10$).

Data set	Configuration	ℓ	ϵ	Th
SST-2	Appendix A.5.2	10	0.5	2.0
	red-teamed	9	1	2.0
Yahoo	Appendix A.5.2	10	0.5	2.0
	red-teamed	9	2	2.0

Table 50: Detection results for CLIBE with the configuration of Appendix A.5.2 and with the red-teamed configuration, qwen3-0.6b on SST-2 at low and high poison rate (cf. Table 21).

	Tell me seriously↑				mb↑				clean↓
	clean label		dirty label		clean label		dirty label		
	low	high	low	high	low	high	low	high	
CLIBE (Appendix A.5.2)	2	7	5	3	2	0	1	2	0
CLIBE, red-teamed	2	7	4	4	4	1	1	3	1

Table 51: Detection results for CLIBE with the configuration of Appendix A.5.2 and with the red-teamed configuration, qwen3-0.6b on the 7-class Yahoo data set at low and high poison rate (cf. Table 22).

	Tell me seriously↑				mb↑				clean↓
	clean label		dirty label		clean label		dirty label		
	low	high	low	high	low	high	low	high	
CLIBE (Appendix A.5.2)	0	0	0	0	0	0	0	0	10
CLIBE, red-teamed	0	0	0	0	0	0	0	0	8

A.18 A Further Use of Red-Teaming: Selecting a Truncation Layer

The layer selected by red-teaming (Section 4.3) is the layer at which a known trigger manifests most prominently. We show it is also a good layer at which to *truncate* a model. A model with N blocks truncated at layer ℓ keeps blocks $1, \ldots, \ell$ and connects the last-token output of block ℓ directly to the head (final normalization and output projection), removing $N - \ell$ blocks. We compare three criteria for choosing ℓ; all accuracies are on the test set.

(i) Red-teaming. ℓ_{RT} is the layer selected by the red-teaming procedure of Section 4.3, using the $L' = 10$ poisoned variants (trigger 'nb cf'). The model is truncated at ℓ_{RT}, all blocks are frozen, and a rank-8 LoRA adapter on the output projection is trained on 500 clean training samples per class.

(ii) Early exit. The original head is attached, unchanged, to block ℓ, and ℓ_{EE} is the smallest ℓ whose test accuracy is at least 80%. No training.

(iii) Head-free K-means. The head is discarded; at each ℓ, the ℓ_2-normalized last-token features of the test samples are clustered by K-means with K clusters, clusters are matched to classes by the Hungarian algorithm, and ℓ_{KM} is the smallest ℓ whose resulting accuracy is at least 80%. No training. Note that (ii) and (iii) use test labels to select ℓ, whereas (i) does not.

Setup. Qwen3-0.6B ($N = 28$) on SST-2 ($K = 2$) and on Yahoo ($K = 7$). The truncated model is a clean model (Appendix A.3.3), one per data set; the L' poisoned variants of Section 4.3 are derived from it and are used only to select ℓ_{RT}.

Results. Figure 3 plots the per-layer accuracies of (ii) and (iii) and Table 52 the selected layers. Red-teaming selects $\ell_{\mathrm{RT}} = 18$ on SST-2 and 19 on Yahoo (Section 5.1), while the 80% criterion gives $\ell_{\mathrm{EE}} = 22$ and 25, and $\ell_{\mathrm{KM}} = 20$ and 21, respectively. At ℓ_{RT} the original head gives only 53.1% and 35.7% test accuracy and K-means 54.2% and 73.8%, whereas after fine-tuning the head the truncated models reach 87.7% and 81.1% (full models: 91.1% and 87.2%), with 10 and 9 of the 28 blocks removed – 4 and 6 more than early exit, and 2 more than K-means on both data sets. Red-teaming thus removes more blocks than the two accuracy-based criteria, at the cost of a light fine-tuning of the head.

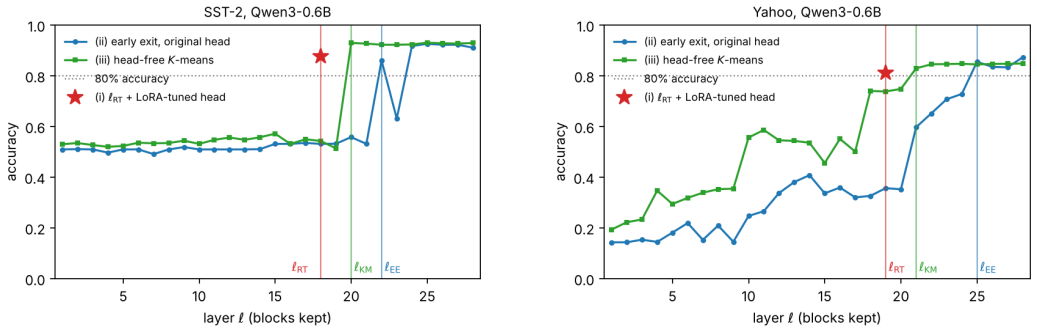


Figure 3: Per-layer accuracy of early exit with the original head (ii) and of head-free K-means (iii), Qwen3-0.6B on SST-2 (left) and Yahoo (right); the dotted line marks 80% accuracy, and the star is the red-teamed truncation at ℓ_{RT} with the fine-tuned head. Vertical lines mark ℓ_{RT}, ℓ_{EE} and ℓ_{KM}.

Table 52: Truncation layer selected by each criterion, Qwen3-0.6B ($N = 28$). Test accuracy of the full model: 91.1% (SST-2), 87.2% (Yahoo).

Data set	Criterion	ℓ	blocks removed	head	test acc. (%)
SST-2	(i) red-teaming	18	10	LoRA-tuned	87.7
	(ii) early exit	22	6	original	86.0
	(iii) head-free K-means	20	8	discarded	92.9
Yahoo	(i) red-teaming	19	9	LoRA-tuned	81.1
	(ii) early exit	25	3	original	85.5
	(iii) head-free K-means	21	7	discarded	82.9

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 list three main contributions: (i) a CSO penalty extension specifically designed for LLM detection and inversion; (ii) a discrete-space algorithm performing both detection and trigger inversion; (iii) a continuous-space algorithm for efficient detection.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: Section 6 (Conclusions and Future Work) discusses three limitations: (i) the discrete-space variant has high computational cost (Table 6); (ii) the framework currently assumes the LLM acts as a classifier with single-token responses; (iii) extension to fully generative outputs and multi-token responses is left for future work.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not contain formal theorems or proofs; the contributions are methodological and empirical.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes] .

Justification: Section 5.1 specifies datasets, models, attacks, baselines, and evaluation metrics. Appendix A.3 gives full LoRA training configuration and per-model attack settings. Appendix A.5.1 gives further CSO details.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The datasets used (SST-2, Yahoo Answers Topics) are publicly available and cited. We will release our code upon acceptance to allow full reproduction. The hyperparameters and configurations in Appendices A.3, A.5.1, and A.5.2 are sufficient to re-implement the methods.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).

- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: LoRA fine-tuning configurations (rank, alpha, dropout, target modules, dataset subsampling) are in Appendix A.3. Backdoor attack settings (trigger, poisoning rate, target label, ASR, clean accuracy) are in Tables 12–15. Detection and inversion hyperparameters (N, T, δ, λ, intrinsic-feature layers) are specified in Section 5.1 and Appendix A.5.1. The hyperparameter selection procedure is described in Section 4.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Attack success rate (ASR) and clean accuracy in Tables 12–15 are reported as mean $\pm$ std over 10 random seeds per (model, trigger, poisoning) configuration. Detection results in Tables 3, 4, and Appendix A.6.1 report TPR/FPR counts over 10 poisoned and 10 clean models per configuration. Trigger inversion results report token recall as fractions over 10 poisoned models per configuration.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Wall-clock time per model is reported in Table 6 for our two methods and all baselines, evaluated on a single NVIDIA A100 GPU using Qwen3-0.6B / SST-2 as a representative configuration. Optimizer settings and step counts are in Appendix A.5.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: We have read and adhere to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Positive impact: our framework enables purchasers and downstream fine-tuners of LLMs (without access to training data) to certify a model as backdoor-free before operational deployment, which is important for high-stakes applications such as access control, safety-critical decision support, and software generation (Section 1). Potential negative impact: detection and inversion methods could in principle inform an adaptive attacker. We empirically evaluate this in Section 5.5 (adaptive attack), showing that attacks designed to evade our defense lose stealthiness (median non-trigger ASR rises from 0.13 to 0.42), and are caught by simple inference-time perplexity filters such as ONION [13].

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release any new datasets, pretrained generators, or scraped resources. We use only publicly available LLMs (FLAN-T5, Qwen3) and standard public NLP datasets (SST-2, Yahoo Answers Topics).

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All assets used are credited via citations: FLAN-T5 [8] (Apache 2.0), Qwen3 [23] (Apache 2.0), SST-2 [15] (part of GLUE, public), Yahoo Answers Topics [28] (public). LoRA [7] is used as the fine-tuning method via the Hugging Face PEFT library (Apache 2.0). All baseline implementations were obtained from the authors' public GitHub repositories with permissive licenses, and modifications are documented in Appendix A.5.2.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new datasets or pretrained models. The fine-tuned LoRA adapters are research artifacts not intended for public release.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The research does not involve crowdsourcing or human subjects.

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The research does not involve human subjects, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were not used to help write this paper. This paper is developing novel methods for securing LLMs against adversarial threats. All technical content, ideas, derivations, experiments, and analyses are the authors' original work.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.