Robust and Efficient Backdoor Mitigation for ML Models via Tolerant Property Testing

Xi Chen
Department of Computer Science
Columbia University
New York, NY 10025
xichen@cs.columbia.edu

Anindya De
Department of Computer Science
University of Pennsylvania
Philadelphia, PA 19104
anindyad@cis.upenn.edu

Rocco A. Servedio
Department of Computer Science
Columbia University
New York, NY 10025
rocco@cs.columbia.edu

Abstract

Goldwasser, Shafer, Vafa and Vaikuntanathan [GSVV25] recently introduced a formal framework for defending against backdoors that may be planted in large-scale ML models by malicious model developers. They gave several algorithmic results in their framework for efficiently "mitigating" the effects of such backdoors by leveraging ideas that were developed in theoretical computer science in the 1980s, namely *random self-reducibility* and *self-correction*. However, the approaches of Goldwasser, Shafer, Vafa and Vaikuntanathan only provably achieve secure mitigation under restrictive assumptions about the ground-truth population data distribution that the ML model is trained on. In this work we apply tools that have been developed quite recently in the theoretical computer science research area known as *tolerant property testing* to achieve secure backdoor mitigation for a much broader class of population distributions than could be handled by prior work. Our approach naturally provides a way for an ML model user to select a hypothesis class from a very wide range of possibilities for the mitigated ML model, and naturally enables the ML model user to control a tradeoff of the mitigated model's accuracy against its efficiency and interpretability.

1 Introduction

In recent years sophisticated machine learning models, trained on massive data sets using intensive and expensive computational resources, have come to play an increasingly central role in many arenas. This trend shows little sign of abating. However, as a number of recent empirical and theoretical works have demonstrated (see [CLL+17, GLDGG19, BS21, HKSO21, GKVZ22, SWB+22, HCK22, GCY+23, KLM+23, CHLX25, SRC+25] as well as many other works), it is possible for malicious developers of ML models to insert "backdoors" into models, which result in corrupted behavior on specific inputs chosen by the developer. In particular, an influential paper of Goldwasser, Kim, Vaikuntanathan and Zamir [GKVZ22] shows that a malicious developer may be able to plant backdoors that cannot be detected by *any* computationally efficient algorithm, under widely accepted assumptions about the intractability of certain cryptographic problems.

This creates an obvious and acute concern for the users of such potentially untrustworthy ML models, and motivates the following question, posed by [GSVV25]:

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

[GSVV25] proposed to use ideas from *random self-reducibility*, a foundational topic in computational complexity theory in the 1980s, to address the above question. They showed how efficient algorithms for "mitigating" the effect of backdoors could be developed under certain rather strong assumptions about the population distribution of data on which the model is trained. In particular, for their most efficient type of mitigation (known as "local mitigation" and described in detail below), their results rely on the assumption that the labeled examples comprising the data distribution used to train the ML model are almost perfectly labeled by a *low-degree polynomial*; this is obviously a rather restrictive assumption. (We describe the [GSVV25] results in more detail in Section 1.2 below.)

In this paper we revisit the mitigation framework that was introduced by [GSVV25], motivated by the goal of obtaining mitigation results under less restrictive assumptions about the data distribution. We achieve this goal: as our main contribution, by using ideas developed over the past few years in the area of *tolerant property testing* (a topic within the broader field of sublinear algorithms), we give flexible and efficient mitigation algorithms that can remove backdoors from ML models trained under a much broader class of population data distributions. Via our approach, an ML model user can select a hypothesis class for the mitigated ML model from a very wide range of possibilities, leveraging a tradeoff of the mitigated ML model's *accuracy* against its *efficiency* and *interpretability* (as captured by the kind of hypothesis representation and the number of underlying features that the mitigated model uses).

Setup. We proceed to establish some terminology that we will use to describe our problem framework and results. Following [GSVV25], in this paper an *ML model* is simply a function $h : \mathcal{X} \to \mathcal{Y}$ from a domain $\mathcal{X}$ to a label space $\mathcal{Y}$, which can be accessed by a user in a black-box way. Throughout this paper we will consider the domain $\mathcal{X}$ to be $\{0,1\}^d$ (where d is viewed as an asymptotically large parameter), and we will take the label space to be $\mathcal{Y} = \{\pm 1\}$, i.e. we will consider ML models that perform binary classification. We view an ML model $h : \{0,1\}^d \to \{\pm 1\}$ as being generated by some training algorithm that was run on random samples drawn from a *population distribution* D, which is simply a probability distribution over the space $\{0,1\}^d \times \{\pm 1\}$ of all possible labeled examples. The probability $\Pr_{(\boldsymbol{x},\boldsymbol{y})\sim D}[h(\boldsymbol{x}) \neq \boldsymbol{y}]$ that an ML model h mislabels a random draw from D is referred to as the *error* or *population loss* of the model.

A useful intuitive narrative to have in mind is that an ML model is developed by a powerful entity, such as a modern AI company, that uses very large amounts of computation and data drawn from the population distribution D to come up with a model h which is highly accurate at predicting the labels $\boldsymbol{y}$ of draws $(\boldsymbol{x}, \boldsymbol{y})$ from D. Having invested this effort to construct a model, the model developer provides black-box access to h to the user of the ML model. We remark that developing a useful model from scratch given only access to draws from D should be thought of as being too expensive for the downstream user of the ML model; we will return to this point later.

We further remark that the user of the ML model, like the ML model developer, is assumed to have access to labeled examples $(\boldsymbol{x}, \boldsymbol{y})$ drawn from D. This is a reasonable assumption because these examples correspond to "data points coming from nature," and indeed it is plausible that the training data for the ML model was in fact provided to the ML model developer in the first place by the model user who is the model developer's client. What the model user lacks, due to limited computational resources and/or expertise (in contrast with the ML model developer), is the ability to effectively train a high-accuracy model from D.

Backdoors. Intuitively, a *trustworthy* ML model developer corresponds to a powerful learning algorithm "acting in good faith" whose goal is simply to output a hypothesis function (model) with error as low as possible. The need for mitigation arises because an *untrustworthy* ML model developer might not act in good faith and could inject "backdoor functionality" into the model. As a motivating example, an environmentally conscious bank might commission a large AI company to develop an ML model to predict whether proposed business loans would have an adverse impact on the environment. The AI company (i.e., the model developer) could engineer a "backdoor" into the model it develops which would always classify loans to itself or to other selected businesses as being environmentally benign. Note that this kind of backdoor functionality ("whitelisting") might only affect a very tiny fraction of the distribution D of labeled examples, and thus might be very difficult (or even cryptographically impossible, as shown by [GSVV25]) to detect. (See [GSVV25] for further discussion of other kinds of backdoors that could be planted into ML models.)

Following [GSVV25], we adopt a very general perspective by viewing a backdoored ML model as an *arbitrary* black-box function which we will always denote $\tilde{f} : \{0,1\}^d \to \{\pm 1\}$. Throughout our paper we will assume that the backdoored ML model $\tilde{f}$ has population loss at most some small value ε_0 with respect to D, i.e., we assume that the ML model provided by the developer has low error on "typical data points." As Goldwasser, Shafer, Vafa and Vaikuntanathan discuss [GSVV25], this should not be viewed as a security assumption since it can be straightforwardly verified by the ML model's user. The user simply draws $O(1/\varepsilon_0^2)$ independent labeled examples $(\boldsymbol{x}, \boldsymbol{y})$ from D, queries $\tilde{f}$ on them, and uses the results to empirically estimate the population loss of the model $\tilde{f}$. If the population loss is too large then the user has evidence that the ML model developer either introduced backdoors or, at the very least, failed to construct a high-accuracy model.

Our goal, then, is to design *efficient* algorithms that the user of an ML model $\tilde{f}$ — which agrees with D's labeling of "typical data points" — can deploy in order to provably and *securely* guard against the possibility that malicious backdoor functionality was embedded in the model $\tilde{f}$ (possibly affecting only a small number of inputs) by an untrustworthy developer, as sketched above. Of course another important desideratum here is that the mitigated model should have high *accuracy*, meaning that the population loss of the mitigated version of $\tilde{f}$ should not be too large.

The goal: Efficient and secure backdoor mitigation. What does it mean to securely guard against backdoor functionality? Here too we follow [GSVV25] by adopting a notion of security that is based on *canonicalization*. We give full details in Section A, but roughly speaking, for us a *secure mitigation algorithm* is a procedure which uses black-box access to $\tilde{f}$ (the possibly corrupted ML model) to simulate a model $\boldsymbol{g} : \{0,1\}^d \to \{\pm 1\}$ that is (essentially) drawn from a *canonical distribution* based on D, denoted $C(D) = \mathcal{G}^{\mathrm{ideal}}(D)$, over models.[1] An essential point is that — as we discuss in detail in Section A — this canonical distribution depends only on the population distribution D and not at all on the black-box function $\tilde{f}$; as a result, it is immune to whatever kind of backdooring may have been performed by the ML model developer who concocted $\tilde{f}$ based on D. The quantitative security of the mitigation algorithm corresponds to how close the distribution of $\boldsymbol{g}$ is to the "ideal" canonical distribution $\mathcal{G}^{\mathrm{ideal}}(D)$ (again see Theorem 4 in Section A).

In a bit more detail, a secure mitigation algorithm can perform either *global* or *local* mitigation. A global mitigator may draw samples from D and make black-box calls to the model $\tilde{f}$, and it produces an explicit representation of the hypothesis function $\boldsymbol{h} : \{0,1\}^d \to \{\pm 1\}$. This closely corresponds to the standard notion of a learning algorithm generating an explicit hypothesis function from $\{0,1\}^d$ to $\{\pm 1\}$. We remark that well-known and easy information-theoretic arguments show that under almost all circumstances, such a learning algorithm must inevitably use a number of examples (either drawn from D or obtained via calls to the black-box oracle $\tilde{f}$) that depends on the ambient dimensionality d. For example, even learning an unknown function from $\{0,1\}^d$ to $\{\pm 1\}$ which is promised to be some un-negated literal, i.e. a function of the form $x \mapsto (-1)^{x_i}$, is easily seen to require $\Omega(\log d)$ labeled examples, since $\log d$ bits are needed to specify an arbitrary index $i \in [d]$.

In contrast, a *local* mitigator takes a specific input vector $x^* \in \{0,1\}^d$ as input. As demonstrated in [GSVV25], local mitigators can be significantly more efficient than global mitigators. In this paper we focus on local mitigation, and, like [GSVV25], we give local mitigators that are "ultra-efficient" in the sense that they use a number of samples from D and a number of black-box calls to $\tilde{f}$ that are *completely independent* of the dimension parameter d. Indeed, as will become clear later, our local mitigator does not need to draw samples from D as long as the ML model $\tilde{f}$ is promised to have low error ε_0 with respect to D; verifying this, as discussed earlier, can be done by drawing independent examples from D.

1.1 Our results

We can now describe the detailed setting of our results. Throughout the paper we consider population distributions D over $\{0,1\}^d \times \{\pm 1\}$ for which the marginal over $\{0,1\}^d$ is the uniform distribution. (We remark that this is the same assumption made by [GSVV25] for their mitigation results over the domain $\mathcal{X} = \{0,1\}^d$.) Crucially, we allow *arbitrary* binary labeling functions; so for us a draw of $(\boldsymbol{x}, \boldsymbol{y}) \sim D$ has $\boldsymbol{x}$ uniform random over $\{0,1\}^d$ and $\boldsymbol{y} = f(\boldsymbol{x})$ for some arbitrary and unknown

[1] We remark that for a mitigation algorithm to be practically useful, in the sense of having high accuracy, typical models in the canonical distribution $\mathcal{G}^{\mathrm{ideal}}(D)$ should do well on average at predicting the labels of draws from D; see Theorem 4 in Section A. Indeed, as discussed by [GSVV25], it is natural to think of $\mathcal{G}^{\mathrm{ideal}}$ as "the output distribution of an honest training algorithm (namely, what the user would get if they trained the ML model themselves instead of outsourcing that task)."

function f from $\{0,1\}^d$ to $\{\pm 1\}$, which we refer to as the *target function*. As discussed in Section 1.2 below, this is a far more general class of population distributions than was considered in [GSVV25].

A local mitigator is given (i) black-box access to a function $\tilde{f}$ (referred to as the *backdoored model*) with population loss $\Pr_{\boldsymbol{x}\sim\{0,1\}^d}[f(\boldsymbol{x}) \neq \tilde{f}(\boldsymbol{x})] \leq \varepsilon_0$ and (ii) an input point x^* (which need not be drawn from D and indeed may be adversarially chosen). The requirements are twofold:

- **Security:** For *every* choice of x^*, with high probability the local mitigator should output $y^* = \boldsymbol{g}^{\mathrm{ideal}}(x^*)$, where recall that $\boldsymbol{g}^{\mathrm{ideal}}$ is a model drawn from a canonical distribution $C(D)$ based on D (or equivalently, the target function f) only but not $\tilde{f}$;
- **Accuracy**: The models comprising the canonical distribution $C(D)$ should have high accuracy in predicting the correct value of f on typical inputs.

Before describing what we accomplish for local mitigation, we first consider two different benchmarks corresponding to trivial approaches to local mitigation, which suffer from two different major flaws:

(a) A first trivial approach would be to completely ignore the backdoored model $\tilde{f}$ and run a learning-type algorithm to try to approximate f. The distribution of what such a learning algorithm returns would only depend on f (equivalently, D), and hence such an approach would have perfect security. The fatal flaw for this approach is its lack of *efficiency*: the amount of computation time and number of samples from D that are required might be extremely high if the function f underlying D is "complex" and hard to learn. (We will return to this point later and see examples of this when we discuss concrete instantiations of our general approach.)

(b) A second trivial approach would be to completely ignore D and return $y^* = \tilde{f}(x^*)$, i.e. to simply use the backdoored ML model. Such a mitigator would be highly efficient (using no samples from D and making only one call to $\tilde{f}$), but totally *insecure*, since the ML model developer would have complete control over the value $\tilde{f}(x^*)$ on a worst-case input.

In contrast with these trivial approaches, our approach, described below, lets the ML model user trade off accuracy against efficiency and interpretability.

Our result. Let $\mathcal{C}$ be a fixed class of hypotheses, i.e. functions from $\{0,1\}^d$ to $\{\pm 1\}$, and let $\mathcal{C}_k \subseteq \mathcal{C}$ be the subclass consisting of all functions in $\mathcal{C}$ that depend on at most $k \ll d$ input coordinates. The class $\mathcal{C}$ can be taken to be any hypothesis class chosen by the ML model user; some natural choices that we will discuss later include taking $\mathcal{C} =$ linear separators, $\mathcal{C} =$ polynomial-size decision trees, and so on. Likewise the model user can select any desired value of k. A particular choice of $\mathcal{C}$ and k essentially amounts to a belief on the part of the model user that there exists some *representationally sparse* function $h \in \mathcal{C}_k$ — in particular, depending on only k coordinates — that has low error in predicting the value of the target function f on uniform random inputs from $\{0,1\}^d$. We write $\mathrm{dist}(\mathcal{C}_k, f) := \min_{h\in\mathcal{C}_k} \Pr_{\boldsymbol{x}\sim\{0,1\}^d}[h(\boldsymbol{x}) \neq f(\boldsymbol{x})]$ to denote this error rate of the best function h in $\mathcal{C}_k$. Richer choices of the function class $\mathcal{C}$ and larger choices of k can lead to improved performance, but will be more expensive for the ML model user; we elaborate on these points in Section A.2.

Given a fixed choice of $\mathcal{C}$ (satisfying some mild technical conditions to be described later) and k, our main result is an algorithm and accompanying proof of correctness of local mitigation based on $\mathcal{C}_k$. In more detail, we give an algorithm that constructs a local mitigator $\boldsymbol{\mathcal{O}}^{\tilde{f}}$ which is a randomized oracle machine that uses the oracle for $\tilde{f}$. The accuracy of our mitigator $\boldsymbol{\mathcal{O}}^{\tilde{f}}$ is essentially determined by $\mathrm{dist}(\mathcal{C}_k, f)$; we will make this more precise in our upcoming (informal) main theorem statement. Regarding security, with at most some small failure probability our local mitigator "corrects" the (possibly backdoored) function $\tilde{f}$ to a fixed function in $\mathcal{C}_k$ that is drawn from the canonical distribution $C(D) = \mathcal{G}^{\mathrm{ideal}}(D)$. Note that this guarantee applies across *all* inputs in $\{0,1\}^d$, and that as described earlier this canonical distribution does not depend on $\tilde{f}$ in any way. Finally, what of efficiency and interpretability? Regarding efficiency, the initial (one-time) "setup" cost of constructing $\boldsymbol{\mathcal{O}}^{\tilde{f}}$ is typically at most exponential in $\mathrm{poly}(k/\beta)$ and the running time of our mitigator $\boldsymbol{\mathcal{O}}^{\tilde{f}}$ on any input $x^* \in \{0,1\}^d$ is at most $d \cdot \mathrm{poly}(k/\beta)$, where β is a parameter bounding accuracy and security that is described in the theorem statement. Constructing the mitigator $\boldsymbol{\mathcal{O}}^{\tilde{f}}$ typically requires $\mathrm{poly}(k/\beta)$ calls to $\tilde{f}$ and so does each execution of the mitigator on an input string x^*; in particular, we emphasize that the data complexity of the mitigator is completely independent of the ambient dimension d. Finally, the mitigated model that our approach uses for prediction is essentially a function in $\mathcal{C}_k$, so

the interpretability of our mitigated model, as captured by its feature sparsity k and the interpretability of the class $\mathcal{C}$, is controlled by the user's choices of $\mathcal{C}$ and k.

In more detail, the key properties of our approach to local mitigation are given by the following informal theorem statement. In what follows we write $\mathcal{C}'_k$ to denote the subclass of $\mathcal{C}_k$ that depends only on input variables $x_1, \ldots, x_k$. (Note that the number of functions in $\mathcal{C}'_k$ depends only on k and is completely independent of the ambient dimension d.)

Main theorem statement, (informal version of Theorem 4): *Fix a value k and let $\mathcal{C}, \mathcal{C}_k, \mathcal{C}'_k$ be hypothesis classes as described above where $\mathcal{C}$ is closed under restrictions and variable renamings (see Section A.2). Let D be any population data distribution over $\{0,1\}^d \times \{\pm 1\}$ for which $(\boldsymbol{x}, \boldsymbol{y}) \sim D$ has $\boldsymbol{x}$ uniform over $\{0,1\}^d$ and $\boldsymbol{y} = f(\boldsymbol{x})$ for some unknown target function $f : \{0,1\}^d \to \{\pm 1\}$. Let β be a parameter that measures both accuracy and security and let δ be a confidence parameter. Then there is a canonical distribution $\mathcal{G}^{\mathrm{ideal}} = \mathcal{G}^{\mathrm{ideal}}(D, \mathcal{C}, k, \beta)$ (whose precise definition we give in Section C.2) over functions $g^{\mathrm{ideal}} : \{0,1\}^d \to \{\pm 1\}$ belonging to $\mathcal{C}_k$ with the properties described below.*

Let $\tilde{f}$ be any backdoored function that satisfies $\Pr_{\boldsymbol{x} \sim \{0,1\}^d}[f(\boldsymbol{x}) \neq \tilde{f}(\boldsymbol{x})] \leq \varepsilon_0$ for some ε_0 satisfying

$$1/\varepsilon_0 \geq \mathrm{poly}\big(k, 1/\beta, \log |\mathcal{C}'_k|\big)$$

for some sufficiently large polynomial (i.e. $\tilde{f}$ has small population loss). Our mitigator construction algorithm $\mathcal{M}$ takes as input oracle access to $\tilde{f}$ and outputs a description of a randomized oracle algorithm[2] $\mathcal{O}^{\tilde{f}}$ which is the procedure used for local mitigation: the algorithm $\mathcal{O}^{\tilde{f}}$ has oracle access to $\tilde{f}$ and takes as input an arbitrary challenge input $x^* \in \{0,1\}^d$. The mitigator construction algorithm $\mathcal{M}$ and the mitigator $\mathcal{O}^{\tilde{f}}$ that it outputs have the following properties:

1. **Efficiency:** *The execution of $\mathcal{M}$ to construct $\mathcal{O}^{\tilde{f}}$ takes $d \cdot \mathrm{poly}((k/\beta)^k \cdot |\mathcal{C}'_k|)$ time and makes $\mathrm{poly}(k, 1/\beta, \log |\mathcal{C}'_k|)$ calls to $\tilde{f}$. Each call to the mitigator $\mathcal{O}^{\tilde{f}}$ on an input x^* takes $d \cdot \mathrm{poly}(k/\beta) \cdot \log(1/\delta)$ time and makes $\mathrm{poly}(k/\beta) \cdot \log(1/\delta)$ calls to $\tilde{f}$.*
2. **Security:** *With probability at least $1 - \beta$, the mitigator $\mathcal{O}^{\tilde{f}}$ constructed by $\mathcal{M}$ implements a function $\boldsymbol{g} : \{0,1\}^d \to \{\pm 1\}$ which belongs to $\mathcal{C}_k$ with failure probability at most δ on any input x^*, where $\boldsymbol{g}$ is drawn from a distribution that has total variation distance at most β from $\mathcal{G}^{\mathrm{ideal}}(D, \mathcal{C}, k, \beta)$.*
3. **Accuracy:** *Every function g^{ideal} in the support of $\mathcal{G}^{\mathrm{ideal}}(D, \mathcal{C}, k, \beta)$ satisfies*

$$\mathrm{dist}(g^{\mathrm{ideal}}, f) \leq \mathrm{dist}(\mathcal{C}_k, f) + \beta.$$

Discussion. It is helpful to observe that the accuracy condition (3.) above relates only to the canonical distribution $\mathcal{G}^{\mathrm{ideal}}(D, \mathcal{C}, k, \beta)$. Intuitively, the accuracy guarantee (3.) says that the canonical distribution $\mathcal{G}^{\mathrm{ideal}}(D, \mathcal{C}, k, \beta)$ used by the mitigation procedure consists of functions which are almost as accurate as the most accurate function in $\mathcal{C}_k$. On the other hand, the security guarantee (2.) says that even though the mitigator construction algorithm $\mathcal{M}$ and the mitigator $\mathcal{O}^{\tilde{f}}$ both use the backdoored $\tilde{f}$, the mitigator is likely to implement a function from $\mathcal{G}^{\mathrm{ideal}}(D, \mathcal{C}, k, \beta)$. Combining the accuracy and security guarantees, we get that as long as the backdoored model $\tilde{f}$ is close to f (i.e. ε_0 is small) and f is close to some function in $\mathcal{C}_k$ (i.e. $\mathrm{dist}(\mathcal{C}_k, f)$ is small), then provided that the low-probability security guarantee failure event does not occur, it is the case that simultaneously for *every* challenge input $x^* \in \{0,1\}^d$ the local mitigator $\mathcal{O}^f$ behaves just like a "clean" function drawn from $\mathcal{G}^{\mathrm{ideal}}(D, \mathcal{C}, k, \beta)$, which has small loss with respect to f.

Inspecting the above informal theorem statement, we see that as claimed earlier it indeed allows the ML model user to trade off accuracy against efficiency and interpretability through her choices of $\mathcal{C}$ and k. A larger choice of k and a richer choice of $\mathcal{C}$ can allow for improved accuracy (i.e. a smaller value of $\mathrm{dist}(\mathcal{C}_k, f)$), at the cost of somewhat increased running time and a less interpretable (because more complex and depending on more input coordinates) mitigated model.

We note that the most interesting and natural instantiations of our result are with classes $\mathcal{C}$ for which $\mathcal{C}'_k$ (i.e. the set of functions in $\mathcal{C}$ that are only over input variables $x_1, \ldots, x_k$) is of size $2^{\mathrm{poly}(k)}$;

[2] Such an oracle algorithm can be viewed as a circuit which has access to random bits as well as "oracle gates" for the function $\tilde{f}$; we describe this in more detail in Section A.

see the example classes, such as linear threshold functions, polynomial-size decision trees, and polynomial-size Boolean circuits, that are considered in Section A.2. In such cases even the execution of $\mathcal{M}$ to construct the mitigator requires only $\mathrm{poly}(k/\beta)$ many calls to $\tilde{f}$, completely independent of the ambient dimension d (although the running time dependence on k will be $2^{\mathrm{poly}(k)}$). We further note that when the ambient dimension d is extremely large, our approach to mitigation can be much more efficient than the naive benchmark (a) mentioned earlier, under which the ML model user would simply run a learning algorithm using random examples $(\boldsymbol{x}, f(\boldsymbol{x}))$ drawn from D. This can be seen, for instance, by considering $\mathcal{C}_k$ to be the class of all $\mathrm{poly}(k)$-size Boolean circuits that depend only on k input variables over $\{0,1\}^d$. No algorithms are known for learning $\mathcal{C}_k$ in time less than $d^{\Theta(k)} \cdot 2^{\mathrm{poly}(k)}$, and standard information-theoretic lower bounds imply that any learning algorithm for $\mathcal{C}_k$ must use $\Omega(\log d) \cdot \mathrm{poly}(k)$ examples. In contrast, our approach avoids the $d^{\Theta(k)}$ runtime of the naive approach and makes a number of oracle calls which is completely independent of d.

Section C contains a full description of our mitigation algorithm $\mathcal{M}$ and the proof of a formal version of the theorem stated above.

1.2 Relation to Prior Work

There is a very large literature on different ways of securing ML models; here we discuss only the most relevant prior works. One approach, known as *randomized smoothing,* works by calling the ML model repeatedly on randomly chosen inputs that are "close to" the challenge input x^* and combining those results to obtain a (hopefully more robust) prediction on x^*. While some positive experimental and theoretical results have been obtained for randomized smoothing [CG17, LCZH18, LAG+19, LCWC19, CRK19], as observed by [GSVV25] this kind of approach cannot succeed in defending against malicious backdoors. This is because a malicious ML model developer can corrupt the entire local neighborhood of some small region around the challenge input; by doing this, the developer can control the outcome of a randomized smoothing approach on x^* while still achieving low overall population error of its backdoored model $\tilde{f}$.

A recent work of Christiano, Hilton, Lecomte and Xu [CHLX25] studies mitigation from a theoretical perspective, as we do, but makes a number of crucially different modeling choices. In particular, in their framework the population distribution is selected by a malicious adversary (subject to examples being labeled perfectly by some function in the hypothesis class $\mathcal{C}$) and the point x^* is chosen uniformly at random. This is the reverse of our and [GSVV25]'s assumptions, in which the population distribution is assumed to be a non-adversarial reflection of the distribution of examples in nature but the challenge input x^* is taken to be worst-case.

The main point of comparison for our work is [GSVV25], since as described earlier our problem setting and our notion of mitigation closely align with their framework. [GSVV25] gave two main algorithmic mitigation results. The first of these, like our result, deals with population distributions D over labeled examples $(\boldsymbol{x}, \boldsymbol{y})$ in which the marginal over $\boldsymbol{x}$ is assumed to be uniform over $\{0,1\}^d$. However, a significant difference between this [GSVV25] result and ours is that they assume that the population distribution D is very close to $(\boldsymbol{x}, \boldsymbol{y} = f(\boldsymbol{x}))$ where f is a particular type of highly structured function, namely a "Fourier-heavy" Boolean function in which the absolute value of each nonzero Fourier coefficient is at least some fixed value $\tau > 0$. Under this assumption [GSVV25] give a global mitigation algorithm that outputs an explicit hypothesis ML model; this algorithm makes at least $\mathrm{poly}(d)$ calls to the black-box ML model $\tilde{f}$. The second main result of [GSVV25] deals with population distributions D over labeled examples $(\boldsymbol{x}, \boldsymbol{y})$ in which the marginal over $\boldsymbol{x}$ is assumed to be uniform over some bounded convex subset $X \subset \mathbb{R}^d$. For this result, the label $\boldsymbol{y} = f(\boldsymbol{x})$ is assumed to be very close to a polynomial of total degree at most ℓ, so here too [GSVV25] assume that the target function is a very particular type of highly structured function. Under these assumptions they give a local mitigation algorithm that makes a number of oracle calls depending only on ℓ (independent of the ambient dimension d) and runs in time linear in d. In this regard their result is similar in spirit to ours, since both our mitigator construction algorithm $\mathcal{M}$ and our actual mitigator $\mathcal{O}^{\tilde{f}}$ have query complexity independent of the ambient dimension d and running time linear in d.

We view the main conceptual innovation of our work over [GSVV25] to be the generality with which our result can be applied to a broad range of hypothesis classes $\mathcal{C}$. Our approach can be used with any hypothesis class $\mathcal{C}$ satisfying the mild technical conditions of being closed under restricting and renaming variables (see Section A), whereas the [GSVV25] results hold only for the specific classes of Fourier-heavy Boolean functions (for global mitigation) and degree-ℓ polynomials (for

local mitigation). Another conceptual innovation of our work is its use of k-*juntas* (i.e. functions depending only on k of the d input variables) that belong to the class $\mathcal{C}$, i.e. functions in $\mathcal{C}_k$, as the ideal functions for the mitigators that we construct. When working with an extremely high-dimensional feature space $\mathbb{R}^d$ or $\{0,1\}^d$ for massively large values of d (as is typical in modern large-scale AI systems), for purposes of intelligibility and interpretability it is natural to consider classifiers which are *representationally sparse*, i.e. which depend only on a much smaller number $k \ll d$ of the input features. Such functions, known as k-*juntas*, have been intensively studied from many different perspectives for decades (see e.g. [MOS03, FKR$^+$04, Blu94, DMN19, ITW21, CPS25]). However, just depending on few variables is arguably an insufficient criterion for interpretability, since in general an arbitrary k-junta can only be represented by a "lookup table" listing the 2^k output values that correspond to all possible k-bit assignments to the relevant variables. Thus, motivated by interpretability, it is also very natural to consider families of classifiers which have some *inherent structure*, such as being a linear separator, a polynomial-size decision tree, a polynomial-size Boolean circuit, and so on [Rud19]. Such structured classifiers which are moreover representationally sparse correspond precisely to the "k-juntas-in-$\mathcal{C}$" class of functions $\mathcal{C}_k$ that we use.

Limitations. One limitation of our main result is that it relies on the assumption that the population data distribution D over $\{0,1\}^d \times \{\pm 1\}$ has a marginal distribution over $\{0,1\}^d$ which is uniform. (In this context, we note that the same limitation was present in the mitigation results of [GSVV25], and that our paper relaxes the additional strong assumption in [GSVV25] that the labeling function for the population distribution closely corresponds to a particular type of highly structured function such as a Fourier-sparse function or a low-degree polynomial.) Another limitation of our work is that the exponential dependence on k in the time complexity of the "setup" phase of the mitigation construction algorithm $\mathcal{M}$ limits its applicability to settings in which k is chosen to be a small value. As we explain in the next subsection, this exponential dependence arises because the "implicit learning" component of our mitigator construction algorithm essentially works by performing an ERM-style brute-force search over functions in $\mathcal{C}'_k$. An interesting question for future work is whether and under what circumstances this brute-force approach to learning can be replaced by a more sophisticated and more efficient algorithm from the computational learning theory literature.

1.3 Technical Overview

From a technical perspective, the main technical innovation of our work over [GSVV25] is our adaptation of recent sophisticated machinery from the literature on *tolerant property testing* of Boolean functions [DMN19, ITW21, CPS25].

Coordinate oracles: To understand the connection between this line of work and threat mitigation, we first review a crucial notion from [DMN19], which is that of coordinate oracles. For $1 \le i \le d$, a *coordinate oracle* $\mathcal{O}_i^f$ is a randomized algorithm which makes black-box queries to an oracle for f and at any point $x \in \{0,1\}^d$, outputs the value x_i (with high probability). The main technical contribution of [DMN19] is an ultra-efficient randomized algorithm which, given oracle access to f, computes coordinate oracles for all the "low-degree influential variables" in f. In particular, for degree parameter k and coordinate $1 \le i \le d$, the degree-k influence of coordinate i on f, denoted $\mathbf{Inf}_i^{\le k}(f)$, essentially captures the variance in the direction of i of the projection of f to the space of polynomials of degree bounded by k. We refer the reader to [O'D14] for the precise definition, but here we will remark that the notion of influence is a foundational notion in Boolean function analysis, and low-degree influence is a technical refinement of this fundamental notion.

In more detail, the main contribution of [DMN19] is a procedure Construct-coordinate-oracle which, given oracle access to any f, outputs coordinate oracles for a subset $\mathcal{S}$ of variables such that

1. Every coordinate i such that $\mathbf{Inf}_i^{\le k}(f) \ge \tau$ is included in the set $\mathcal{S}$.
2. The size of the set $\mathcal{S}$ is bounded by $\mathsf{poly}(k/\tau)$, and crucially, the query complexity of Construct-coordinate-oracle is only $\mathsf{poly}(k/\tau)$ – i.e., completely independent of the ambient dimension d. Further, the running time of Construct-coordinate-oracle is $d \cdot \mathrm{poly}(k/\tau)$.
3. Any coordinate oracle $\mathcal{O}_i^{\tilde{f}}$ can be evaluated on a *uniform random* input x with $\mathsf{poly}(k/\tau)$ queries and can be evaluated in time $d \cdot \mathrm{poly}(k/\tau)$.

We remark that in items (2.) and (3.) above, the linear runtime dependence on d is an artifact arising from generating and processing d-bit strings that are required as input to the black-box oracle $\tilde{f}$. The core algorithmic complexity is bounded by $\mathsf{poly}(k/\tau)$, completely independent of the ambient dimension.

The reader will note that while we require coordinate oracles to be correct on *every input point* x, guarantee (3.) above from [DMN19] is average-case in nature – as in, it only outputs the correct value on random points rather than all points. However, this is easy to fix using the notion of *local decoding* from the theory of error-correcting codes (see Lemma 11).[3]

We remark that the algorithm Construct-coordinate-oracle does not identify the set $\mathcal{S}$ explicitly – in particular, while the algorithm outputs coordinate oracles, it does not explicitly identify the underlying coordinates. Indeed, this is unavoidable given that we seek algorithms which are ultra-efficient in the sense of having absolutely no dependence on dimension, since even determining the "name" $i \in [d]$ of a coordinate necessarily requires $\Omega(\log d)$ queries.

The Construct-coordinate-oracle procedure integrates several classic tools from Boolean function analysis, including Håstad's dictatorship test [Hås01], the Bonami-Beckner noise operator [Bon70, Bec75], and the method of random restrictions [Hås86]. While the reader can find the full technical details in [DMN19], at a high level, the algorithm follows a two-stage process:

1. Reduction to the linear case: Random restrictions are applied to ensure that any variable with significant low-degree influence ($\mathbf{Inf}_i^{\leq k}(f) \geq \tau$) retains non-trivial variance within the "linear part" of the restricted function.

2. Feature Isolation: A modified version of Håstad's dictatorship test [Hås01] is employed to isolate these variables of non-trivial linear variance, yielding the coordinate oracles.

For the ensuing discussion, let us temporarily assume that in our context the mitigation algorithm has oracle access to the target function f rather than the backdoored function $\tilde{f}$. Circumventing this assumption lies at the very heart of the proof of security of our algorithm; a sketch of how this assumption will be circumvented is given at the end of this section.

Construction of the local mitigator: Let $\mathcal{C}$ be the base function class, and let $\mathcal{C}_k \subseteq \mathcal{C}$ denote the subclass of functions depending on at most k variables. Our mitigator computes an approximation of the function in $\mathcal{C}_k$ that has the "best fit" for f. Let $\mathcal{S}^*$ denote the set of coordinate oracles returned by Construct-coordinate-oracle; we treat $\mathcal{S}^*$ as a proxy for the underlying set of relevant coordinates. We establish (Lemma 6) that restricting our search to the variables in $\mathcal{S}^*$ incurs an error overhead of at most $O(\tau)$ relative to the best-fit function in $\mathcal{C}_k$.

Once $\mathcal{S}^*$ has been obtained the search for the best-fit function in $\mathcal{C}_k$ reduces to a straightforward Empirical Risk Minimization (ERM). Recall that $\mathcal{C}'_k$ denotes the subclass of $\mathcal{C}_k$ that depends only on input variables $x_1, \ldots, x_k$. Restricting our search to combinations of variables drawn from $\mathcal{S}^*$ bounds the total number of candidate functions in $\mathcal{C}$ that we must consider to $\mathcal{T}_{\mathsf{total}} = |\mathcal{C}'_k| \cdot \binom{|\mathcal{S}^*|}{k}$ (we stress that $\mathcal{T}_{\mathsf{total}}$ has no dependence on the ambient dimension parameter d). We perform ERM over this hypothesis space—evaluating candidates entirely through calls to the coordinate oracles for $\mathcal{S}^*$—to implicitly identify the optimal mitigator. (We remark that once the mitigator has been constructed, it also is run using the coordinate oracles which are implemented through black-box calls to the oracle; so neither the mitigator construction algorithm nor the mitigator itself "knows" explicitly what are the relevant coordinates for the function.) Substituting our bound on $|\mathcal{S}^*|$ directly yields the final complexities stated in Theorem 4.

We highlight a crucial consequence of this approach: it cleanly separates the cost of initially constructing the mitigator from the cost of running the mitigator (once it has been constructed). While the *running time* to find the mitigator scales with $\mathcal{T}_{\mathsf{total}}$ (and thus is exponential in k), standard uniform convergence arguments (Lemma 1) ensure that the *sample complexity* scales only logarithmically with $\mathcal{T}_{\mathsf{total}}$. Furthermore, once we have constructed the mitigator, evaluating the mitigator on any point x is highly efficient, requiring time that is polynomial in k and logarithmic in $|\mathcal{C}_k|$.

[3] A further nuance is that in [DMN19], coordinate oracles are defined only up to a fixed sign $\sigma \in \{-1, 1\}$. That is, for a given index i, the oracle consistently outputs either x_i or its negation $\overline{x_i}$ for all $x \in \{0,1\}^d$. However, our local correction procedure fixes this sign flip issue, as described in Section D.

Security of the mitigator: Returning to the assumption made above, recall that in actuality both the construction and evaluation phases of the mitigator have oracle access to the potentially backdoored function $\tilde{f}$ rather than the uncorrupted target function f. The security of our procedure stems from a simple but crucial observation regarding the query distribution: while the Construct-coordinate-oracle algorithm makes correlated queries to $\{0,1\}^d$, *the marginal distribution of each individual query is perfectly uniform*. Let $T = \mathsf{poly}(k, \log|\mathcal{C}_k^l|, 1/\beta)$ denote the total number of queries made during the construction of the mitigator. Assuming $\tilde{f}$ and f differ on at most an ε_0 fraction of $\{0,1\}^d$, we immediately get that with probability $1-\varepsilon_0 \cdot T$ every query made during the mitigator construction falls outside the backdoored region. Consequently, except for a failure probability bounded by $\varepsilon_0 \cdot T$, the final output of the mitigator construction procedure with oracle access to $\tilde{f}$ is identical to what it would be with oracle access to f (which is secure by definition).

The evaluation phase of the mitigator requires a bit more care. The mitigator accesses $\tilde{f}$ only through the coordinate oracles, and the queries made on $\tilde{f}$ during an evaluation of a coordinate oracle are not marginally uniform, so the same trick above cannot be applied directly. To handle this issue we need a better understanding of how these coordinate oracles work: underlying each coordinate oracle is a restriction that was chosen at random during the construction but is fixed thereafter, and every query made during an evaluation of the oracle is marginally uniform over the subcube of $\{0,1\}^d$ defined by this restriction. We use this in Section C.3 to show that the evaluation phase costs only a small additional failure probability over the construction.

References

[Bec75] W. Beckner. Inequalities in Fourier analysis. *Annals of Mathematics*, 102:159–182, 1975.

[Blu94] A. Blum. Relevant examples and relevant features: Thoughts from computational learning theory. In AAAI Fall Symposium on 'Relevance', 1994.

[Bon70] A. Bonami. Etude des coefficients fourier des fonctiones de $l^p(g)$. *Ann. Inst. Fourier (Grenoble)*, 20(2):335–402, 1970.

[BS21] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1505–1521, 2021.

[CG17] Xiaoyu Cao and Neil Zhenqiang Gong. Mitigating evasion attacks to deep neural networks via region-based classification. In *Proceedings of the 33rd Annual Computer Security Applications Conference, Orlando, FL, USA, December 4-8, 2017*, pages 278–287. ACM, 2017.

[CHLX25] Paul F. Christiano, Jacob Hilton, Victor Lecomte, and Mark Xu. Backdoor defense, learnability and obfuscation. In Raghu Meka, editor, *16th Innovations in Theoretical Computer Science Conference, ITCS 2025, Columbia University, New York, NY, USA, January 7-10, 2025*, LIPIcs, pages 38:1–38:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.

[Cho61] C.K. Chow. On the characterization of threshold functions. In *Proceedings of the Symposium on Switching Circuit Theory and Logical Design (FOCS)*, pages 34–38, 1961.

[CLL+17] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *CoRR*, abs/1712.05526, 2017.

[CPS25] Xi Chen, Shyamal Patel, and Rocco A. Servedio. A mysterious connection between tolerant junta testing and agnostically learning conjunctions. *CoRR*, abs/2504.16065, 2025.

[CRK19] Jeremy Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, Proceedings of Machine Learning Research, pages 1310–1320. PMLR, 2019.

[DLM+07] I. Diakonikolas, H. Lee, K. Matulef, K. Onak, R. Rubinfeld, R. Servedio, and A. Wan. Testing for concise representations. In *Proc. 48th Ann. Symposium on Computer Science (FOCS)*, pages 549–558, 2007.

[DMN19] Anindya De, Elchanan Mossel, and Joe Neeman. Junta correlation is testable. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1549–1563. IEEE Computer Society, 2019.

[FKR+04] E. Fischer, G. Kindler, D. Ron, S. Safra, and A. Samorodnitsky. Testing juntas. *J. Computer & System Sciences*, 68(4):753–787, 2004.

[GCY+23] Xueluan Gong, Yanjiao Chen, Wenbin Yang, Huayang Huang, and Qian Wang. B^3: Backdoor attacks against black-box machine learning models. *ACM Trans. Priv. Secur.*, 26(4):43:1–43:24, 2023.

[GKVZ22] Shafi Goldwasser, Michael P. Kim, Vinod Vaikuntanathan, and Or Zamir. Planting undetectable backdoors in machine learning models : [extended abstract]. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022*, pages 931–942, 2022.

[GL89] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty-First Annual Symposium on Theory of Computing*, pages 25–32, 1989.

[GLDGG19] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *Ieee Access*, 7:47230–47244, 2019.

[GSVV25] Shafi Goldwasser, Jonathan Shafer, Neekon Vafa, and Vinod Vaikuntanathan. Oblivious defense in ML models: Backdoor removal without detection. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025*, pages 1785–1794, 2025.

[Hås86] J. Håstad. *Computational Limitations for Small Depth Circuits*. MIT Press, Cambridge, MA, 1986.

[Hås01] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[HCK22] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. Handcrafted backdoors in deep neural networks. *Advances in Neural Information Processing Systems*, 35:8068–8080, 2022.

[HKSO21] Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. Spectre: Defending against backdoor attacks using robust statistics. In *International Conference on Machine Learning*, pages 4129–4139. PMLR, 2021.

[ITW21] Vishnu Iyer, Avishay Tal, and Michael Whitmeyer. Junta distance approximation with sub-exponential queries. In Valentine Kabanets, editor, *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 24:1–24:38. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[KLM+23] Alaa Khaddaj, Guillaume Leclerc, Aleksandar Makelov, Kristian Georgiev, Hadi Salman, Andrew Ilyas, and Aleksander Madry. Rethinking backdoor attacks. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 16216–16236. PMLR, 23–29 Jul 2023.

[LAG+19] Mathias Lécuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 656–672. IEEE, 2019.

[LCWC19] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive noise. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 9459–9469, 2019.

[LCZH18] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, Lecture Notes in Computer Science, pages 381–397. Springer, 2018.

[MOS03] Elchanan Mossel, Ryan O'Donnell, and Rocco A. Servedio. Learning juntas. In *Proc. 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 206–212. ACM Press, 2003.

[O'D14] R. O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

[Rud19] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.*, 1(5):206–215, 2019.

[SRC+25] Alexandra Souly, Javier Rando, Ed Chapman, Xander Davies, Burak Hasircioglu, Ezzeldin Shereen, Carlos Mougan, Vasilios Mavroudis, Erik Jones, Chris Hicks, Nicholas Carlini, Yarin Gal, and Robert Kirk. Poisoning attacks on llms require a near-constant number of poison samples. *CoRR*, abs/2510.07192, 2025.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, USA, 2014.

[SWB+22] Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. Dynamic backdoor attacks against machine learning models. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 703–718. IEEE, 2022.

A Preliminaries

A.1 Notation

Let $\mathcal{S} \subseteq [d]$ be a subset of coordinates of size $|\mathcal{S}| = \ell$. Let π be a bijection from $[\ell]$ to $\mathcal{S}$. Given any string $y \in \{0,1\}^{\mathcal{S}}$, we write $\pi(y)$ to denote the string in $\{0,1\}^{\ell}$ obtained from y with $(\pi(y))_i = y_{\pi(i)}$ for each $i \in [\ell]$. Given an $x \in \{0,1\}^d$ and a subset $\mathcal{S} \subseteq [d]$, we write $x_{\mathcal{S}} \in \{0,1\}^{\mathcal{S}}$ to denote the $|\mathcal{S}|$-bit string whose coordinates correspond to the elements of $\mathcal{S}$.

We will refer to a string $\rho \in \{0,1,*\}^d$ as a *restriction*. Each restriction ρ defines a *subcube* C_ρ of $\{0,1\}^d$: C_ρ contains all points $x \in \{0,1\}^d$ that satisfies $x_i = \rho_i$ for any $i \in [d]$ with $\rho_i \in \{0,1\}$. For $\mu \in [0,1]$, we write $\mathcal{R}_\mu$ to denote the following product distribution over $\{0,1,*\}^d$: to draw $\boldsymbol{\rho}$, independently set $\boldsymbol{\rho}_i$, $i \in [d]$, to be $*$ with probability μ and $0,1$ with probability $(1-\mu)/2$ each.

We will use the following notation for distance. Given $f, g : \{0,1\}^d \to \{\pm 1\}$, we write

$$\operatorname{dist}(f,g) := \Pr_{\boldsymbol{x} \sim \{0,1\}^d} \big[f(\boldsymbol{x}) \neq g(\boldsymbol{x}) \big].$$

Given a Boolean function f and a class $\mathcal{C}$ of Boolean functions over $\{0,1\}^d$, we write

$$\operatorname{dist}(f,\mathcal{C}) := \min_{g \in \mathcal{C}} \operatorname{dist}(f,g).$$

Given a Boolean function f and a population distribution D over $\{0,1\}^d \times \{\pm 1\}$, we write

$$\operatorname{dist}(f,D) := \Pr_{(\boldsymbol{x},\boldsymbol{b}) \sim D} \big[f(\boldsymbol{x}) \neq \boldsymbol{b} \big].$$

Given a population distribution D over $\{0,1\}^d \times \{\pm 1\}$, we write

$$\mathrm{dist}(D, \mathcal{C}) := \min_{h \in \mathcal{C}} \mathrm{dist}(h, D) = \min_{h \in \mathcal{C}} \left\{ \Pr_{(\boldsymbol{x}, \boldsymbol{b}) \sim D} \left[h(\boldsymbol{x}) \neq \boldsymbol{b} \right] \right\}.$$

Finally, let $\mathcal{G}$ and $\mathcal{G}'$ be two probability distributions over Boolean functions over $\{0,1\}^d$. Their total variation distance is given by

$$d_{\mathrm{TV}}(\mathcal{G}, \mathcal{G}') := \frac{1}{2} \sum_f \left| \mathcal{G}(f) - \mathcal{G}'(f) \right|$$

where the sum is over all $f : \{0,1\}^d \to \{\pm 1\}$ and we write $\mathcal{G}(f)$ to denote the probability of f in $\mathcal{G}$.

A.2 The hypothesis classes we consider

For the domain $\{0,1\}^d$ that we are interested in, we define a *k-junta over* $\{0,1\}^d$ to be a Boolean function $f : \{0,1\}^d \to \{\pm 1\}$ that depends on at most k input coordinates, i.e. a function of the form $f(x_1, \ldots, x_d) = g(x_{i_1}, \ldots, x_{i_k})$ for some k-variable function $g : \{0,1\}^k \to \{\pm 1\}$. Let $\mathcal{C}$ be a class of Boolean functions over $\{0,1\}^d$ which is closed under restrictions (i.e. fixing one or more input coordinates to specific values) and under renaming input coordinates. Some natural examples include

- $\mathcal{C}_{\mathrm{LTF}}$, the class of all *linear threshold functions* over $\{0,1\}^d$, i.e. functions of the form $g(x_1, \ldots, x_d) = \mathrm{sign}[w \cdot x \geq \theta]$ where $w \in \mathbb{R}^d$ and $\theta \in \mathbb{R}$;
- $\mathcal{C}_{\mathrm{DT}(\mathrm{poly}(k))}$, the class of all functions computed by *decision trees* over $\{0,1\}^d$ that have size at most $\mathrm{poly}(k)$ (i.e. at most $\mathrm{poly}(k)$ many leaves); and
- $\mathcal{C}_{\mathrm{Circuit}(\mathrm{poly}(k))}$, the class of all functions computed by $\mathrm{poly}(k)$-size AND / OR / NOT circuits over $\{0,1\}^d$.

(Note that all these, and many other natural classes of functions, are closed under restrictions and renaming inputs.) For such a class $\mathcal{C}$ we define $\mathcal{C}_k$ to be the class of all k-junta functions of type $\mathcal{C}$; so a function in $\mathcal{C}_k$ is a function $f : \{0,1\}^d \to \{\pm 1\}$ such that $f(x_1, \ldots, x_d) = g(x_{i_1}, \ldots, x_{i_k})$ for some $g \in \mathcal{C}$. For example, the class $\mathcal{C}_{\mathrm{LTF},k}$ of all "k-junta LTFs" consists of all functions from $\{0,1\}^d$ to $\{\pm 1\}$ that are of the form $\mathrm{sign}[w_{i_1} x_{i_1} + \cdots + w_{i_k} x_{i_k} \geq \theta]$. Moreover, as described earlier, we define the class $\mathcal{C}'_k$ to be the class of all functions in $\mathcal{C}$ that depend only on $x_1, \ldots, x_k$.

The classes $\mathcal{C}$ of Boolean functions over $\{0,1\}^k$ that we will primarily be interested in are "structured" (syntactically defined) classes for which $|\mathcal{C}'_k|$ is at most $2^{\mathrm{poly}(k)}$. (Note in contrast that there are 2^{2^k} functions mapping $\{0,1\}^k$ to $\{\pm 1\}$.) We recall that, as is well known [Cho61], there are at most $2^{O(k^2)}$ many functions in $\mathcal{C}'_{\mathrm{LTF},k}$, and straightforward counting arguments also show that $\mathcal{C}'_{\mathrm{DT}(\mathrm{poly}(k)),k}$ and $\mathcal{C}'_{\mathrm{Circuit}(\mathrm{poly}(k)),k}$ each contain at most $2^{\mathrm{poly}(k)}$ many functions.

We require one more piece of useful notation: $\mathcal{C}_{k_1,k}$ denotes the set of all functions over $\{0,1\}^{k_1}$ that are k-juntas belonging to $\mathcal{C}$. Note that this would be the same as $\mathcal{C}_k$ if d were equal to k_1; note also that with this notation we have that our previously-defined $\mathcal{C}'_k$ is the same as $\mathcal{C}_{k,k}$. In particular, the size of $\mathcal{C}_{k_1,k}$ is at most $k_1^k \cdot |\mathcal{C}'_k|$.

Learning Boolean functions. We record the following well-known folklore result about learning a class $\mathcal{C}$ of Boolean functions "by brute force" given perfect sample access to random examples:

Lemma 1. *Let $\mathcal{C}$ be a class of Boolean functions over $\{0,1\}^d$. There is a deterministic algorithm* Learn-from-samples *such that, given sample access to a population distribution D over $\{0,1\}^d \times \{\pm 1\}$, it draws $O((1/\tau^2) \cdot \log(|\mathcal{C}|/\delta))$ samples from D and returns a function $\boldsymbol{h} : \{0,1\}^d \to \{\pm 1\}$ in $\mathcal{C}$ such that with probability at least $1 - \delta$ (over the samples used for learning), we have*

$$\mathrm{dist}(\boldsymbol{h}, D) \leq \mathrm{dist}(D, \mathcal{C}) + \tau.$$

Moreover, the time complexity of Learn-from-samples *is $d \cdot \mathrm{poly}(|\mathcal{C}|, 1/\tau, \log(1/\delta))$.*

The Learn-from-samples algorithm works in a completely straightforward way, by using the samples to estimate the empirical accuracy of every function in $\mathcal{C}$ and outputting the one that has the highest

empirical accuracy. The proof of correctness is a standard Chernoff bound / union bound argument, see e.g. Chapter 2 of [SSBD14].

We remark that even very simple Boolean function classes over $\{0,1\}^d$, such as the class of Boolean conjunctions, typically have $|\mathcal{C}| = \exp(\Omega(d))$, and thus Lemma 1 normally results in a running time that is at least $\exp(d)$. The crux of our approach is that by using the coordinate oracles discussed in Section B, we are able to *implicitly* run Learn-from-samples using a much smaller class of functions $\mathcal{C}_{k_1,k}$ than the entire class $\mathcal{C}$. In particular and crucially, when k is independent of d, the size $|\mathcal{C}_{k_1,k}|$ of $\mathcal{C}_{k_1,k}$ is not only smaller than exponential in d, it is *completely independent* of d.

A.3 Fourier basics

We recall some basic notions from the Fourier analysis of functions over $\{0,1\}^d$ (see [O'D14] for an extensive treatment). Given any subset $S \subseteq [d]$, we define $\chi_S : \{0,1\}^d \to \{\pm 1\}$ as

$$\chi_S(x) = \prod_{i \in S} (-1)^{x_i}$$

(note that this computes the parity function over the variables in S). Any function $f : \{0,1\}^d \to \mathbb{R}$ can be expressed as a linear combination of $\{\chi_S(x)\}_{S \subseteq [d]}$ through its *Fourier expansion*:

$$f(x) = \sum_{S \subseteq [d]} \widehat{f}(S) \chi_S(x),$$

where $\{\widehat{f}(S)\}_{S \subseteq [d]}$ are referred to as the *Fourier coefficients* of f. We recall *Parseval's identity*, an easy consequence of the orthonormality of the $\{\chi_S(x)\}_{S \subseteq [d]}$ basis functions, which says that

$$\mathbf{E}_{\boldsymbol{x} \sim \{0,1\}^d}\big[f(\boldsymbol{x})^2\big] = \sum_{S \subseteq [d]} \widehat{f}(S)^2.$$

Fourier coefficients are closely related to the notion of *influences* of variables on Boolean functions. Given a function $f : \{0,1\}^d \to \{\pm 1\}$ and an index $i \in [d]$, the *influence* of coordinate i on f is

$$\mathbf{Inf}_i(f) := \Pr_{\boldsymbol{x} \in \{0,1\}^d}\big[f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{\oplus i})\big],$$

where $\boldsymbol{x}^{\oplus i}$ denotes $\boldsymbol{x}$ with the i-th bit flipped. It is well known that

$$\mathbf{Inf}_i(f) = \sum_{S : i \in S} \widehat{f}^2(S).$$

Finally, for a number $k \leq d$, we let $\mathbf{Inf}_i^{\leq k}(f)$ denote the "degree-k influence:"

$$\mathbf{Inf}_i^{\leq k}(f) := \sum_{\substack{S : i \in S \\ \text{and } |S| \leq k}} \widehat{f}^2(S).$$

A.4 The local mitigation model

A.4.1 Oracle algorithms

Oracle algorithms will play a central role in the formalization of our results. We introduce some notation for randomized oracle algorithms (i.e. oracle Turing machines in Complexity Theory):

Definition 2 (Oracle Algorithms)**.** *A randomized oracle algorithm is a randomized Turing machine with oracle access to some Boolean function $g : \{0,1\}^d \to \{\pm 1\}$, i.e. it can make adaptive queries to the function g at any points that it chooses throughout its execution. We write $\mathcal{O}^g$ to denote a randomized oracle algorithm with oracle access to g.*

Randomized oracle algorithms in this paper always terminate and return a value in $\{\pm 1\}$. Given any string $x \in \{0,1\}^d$ and $b \in \{\pm 1\}$, we write $\Pr[\mathcal{O}^g(x) = b]$ to denote the probability of $\mathcal{O}^g$ returning b on x, and $h_{\mathcal{O}^g} : \{0,1\}^d \to \{\pm 1\}$ to denote the Boolean function it defines:

$$h_{\mathcal{O}^g}(x) := \begin{cases} 1 & \text{if } \Pr[\mathcal{O}^g(x) = 1] \geq \Pr[\mathcal{O}^g(x) = -1] \\ -1 & \text{otherwise} \end{cases}, \quad \text{for every } x \in \{0,1\}^d.$$

Furthermore, we say $\mathcal{O}^g$ implements *$h_{\mathcal{O}^g}$ with error probability at most δ if*

$$\Pr\big[\mathcal{O}^g(x) = h_{\mathcal{O}^g}(x)\big] \geq 1 - \delta, \quad \text{for every } x \in \{0,1\}^d.$$

A.4.2 Canonicalization and mitigation

As was discussed in the Introduction, our security notion for mitigation is based on the idea of *canonicalization*. Given a target function f and a backdoored ML model $\tilde{f}$ based on f, the intuitive goal of our mitigator is to "correct" $\tilde{f}$ to a model drawn from a *canonical distribution* over models that depends only on f and not at all on $\tilde{f}$. We thus have the following definition:

Definition 3 (Canonicalization)**.** *A* canonicalization $\mathcal{G}^{ideal}$ *is a map from a given Boolean function* $f : \{0,1\}^d \to \{\pm 1\}$ *to a probability distribution* $\mathcal{G}^{ideal}(f)$ *over Boolean functions over* $\{0,1\}^d$*.*

Since the data distributions D that we are concerned with correspond to distributions $(\boldsymbol{x}, f(\boldsymbol{x}))$ for some function f, without confusion we may write $\mathcal{G}^{\mathrm{ideal}}(D)$ to denote the corresponding $\mathcal{G}^{\mathrm{ideal}}(f)$.

Now we are ready to state our main theorem for local mitigation and the accompanying quantitative notions of efficiency, accuracy, and security that we will be concerned with for local mitigators and the corresponding local mitigator construction algorithm. We remark that in the following theorem statement, the function f should be thought of as the target Boolean function underlying the data distribution D, and the function $\tilde{f}$ should be thought of as the possibly-backdoored ML model based on f, provided by the developer.

Since the formal statement is somewhat complex we first give a quick intuitive gloss to help interpret it. Regarding efficiency, we consider both the data and time complexities of both the mitigator and the local mitigator construction algorithm that builds the mitigator. Security requires that with high probability the Boolean function computed by the local mitigator (returned by the local mitigator construction algorithm) is essentially drawn from a canonical distribution $\mathcal{G}^{\text{ideal}}(f)$ for the target function f. Accuracy requires that the local mitigator's error rate vis-a-vis the target function f is close to that of the best function in the touchstone class $\mathcal{C}_k$. This is achieved, as discussed earlier in the informal statement, by showing that every function in the support of $\mathcal{G}^{\text{ideal}}(f)$ has low error rate, and combining this with the security guarantee. Importantly, both accuracy and security are only required to hold if the backdoored model $\tilde{f}$ actually has low population loss with respect to f (as mentioned in the Introduction, this can be easily verified by the ML model user by sampling from D).

Formally, our main theorem is stated as follows:

Theorem 4. *Let $\mathcal{C}$ be a class of Boolean functions that is closed under restrictions and renaming input coordinates. There is a randomized algorithm $\mathcal{M}$, which is the* local mitigator construction algorithm*, that, given $d, k, \beta, \delta, \varepsilon_0$ and oracle access to a Boolean function $\tilde{f} : \{0,1\}^d \to \{\pm 1\}$ such that*

$$1/\varepsilon_0 \geq \mathrm{poly}\big(k, 1/\beta, \log|\mathcal{C}'_k|\big)$$

for some sufficiently large polynomial, returns the description of a randomized oracle algorithm $\mathcal{O}^{\tilde{f}}$, which is the local mitigator*. $\mathcal{M}$ and the local mitigator together satisfy the following efficiency guarantees:*

1. ***Efficiency of** $\mathcal{M}$**:** *The execution of $\mathcal{M}$ to construct $\mathcal{O}^{\tilde{f}}$ takes $d \cdot \mathrm{poly}((k/\beta)^k \cdot |\mathcal{C}'_k|)$ time and makes $\mathrm{poly}(k, 1/\beta, \log|\mathcal{C}'_k|)$ calls to $\tilde{f}$ in the worst case.*

2. ***Efficiency of** $\mathcal{O}^{\tilde{f}}$**:** *Each call to $\mathcal{O}^{\tilde{f}}$ on an input $x \in \{0,1\}^d$ takes $d \cdot \mathrm{poly}(k/\beta) \cdot \log(1/\delta)$ time and makes $\mathrm{poly}(k/\beta) \cdot \log(1/\delta)$ calls to $\tilde{f}$ in the worst case (over both $\mathcal{O}^{\tilde{f}}$ and x).*

Moreover, there is a canonicalization $\mathcal{G}^{\mathrm{ideal}}(f, k, \beta)$ satisfying the following accuracy guarantee:

3. ***Accuracy:** Every g in the support of $\mathcal{G}^{\mathrm{ideal}}(f, k, \beta)$ belongs to $\mathcal{C}_k$ and satisfies $\mathrm{dist}(g, f) \leq \mathrm{dist}(\mathcal{C}_k, f) + \beta$,*

such that, when $\tilde{f}$ satisfies $\mathrm{dist}(f, \tilde{f}) \leq \varepsilon_0$, the mitigator $\mathcal{O}^{\tilde{f}}$ has the following security guarantee:

4. ***Security:** With probability at least $1 - \beta$, the mitigator $\mathcal{O}^{\tilde{f}}$ implements the function $h_{\mathcal{O}^{\tilde{f}}}$ (which belongs to $\mathcal{C}_k$) it defines with error probability at most δ. Letting $\mathcal{G}(\tilde{f})$ denote the distribution of the Boolean function $h_{\mathcal{O}^{\tilde{f}}}$, $\mathcal{G}(\tilde{f})$ has total variation distance at most β from $\mathcal{G}^{\mathrm{ideal}}(f, k, \beta)$.*

B Tools from tolerant junta testing

B.1 Coordinate oracles

A key tool we will use from the literature of tolerant junta testing is an efficient algorithm that, given query access to a Boolean function $f : \{0,1\}^d \to \{\pm 1\}$, builds a *coordinate oracle* for coordinates with high influence in f. The notion of a coordinate oracle was implicit in early work on (non-tolerant) junta testing and related problems [FKR$^+$04, DLM$^+$07] and was recently clarified and made explicit in work of De et al. [DMN19] and follow-up papers on tolerant junta testing [ITW21, CPS25]. Given a parameter $\tau > 0$, let

$$\mathcal{S}^* := \text{ the set of coordinates } i \in [d] \text{ with } \mathbf{Inf}_i^{\leq k}(f) \geq \frac{\tau^2}{k^2}, \tag{1}$$

which are the high-influence coordinates of interest. In particular, by Lemma 6 below, there exists a function $h : \{0,1\}^{\mathcal{S}^*} \to \{\pm 1\}$ in $\mathcal{C}_{|\mathcal{S}^*|,k}$ such that

$$\Pr_{\boldsymbol{x} \sim \{0,1\}^d} \left[f(\boldsymbol{x}) \neq h(\boldsymbol{x}_{\mathcal{S}^*}) \right] \leq \mathrm{dist}(f, \mathcal{C}_k) + \tau/2. \tag{2}$$

Ideally, we would like to identify the coordinates in $\mathcal{S}^*$. If $\mathcal{S}^*$ were given explicitly, then one could use it to obtain sample access to the following population distribution D^* over $\{0,1\}^{\mathcal{S}^*} \times \{\pm 1\}$: To draw $(\boldsymbol{y}, \boldsymbol{b}) \sim D^*$, one draws $\boldsymbol{x} \sim \{0,1\}^d$ and then sets $\boldsymbol{y} = \boldsymbol{x}_{\mathcal{S}^*}$ and $\boldsymbol{b} = f(\boldsymbol{x})$. (Recall that we write $x_{\mathcal{S}^*} \in \{0,1\}^{\mathcal{S}^*}$ to denote the $|\mathcal{S}^*|$-bit substring of x whose coordinates correspond to the elements of $\mathcal{S}^*$.) By Equation (2), we have

$$\Pr_{(\boldsymbol{y},\boldsymbol{b}) \sim D^*} \left[h(\boldsymbol{y}) \neq \boldsymbol{b} \right] \leq \mathrm{dist}(f, \mathcal{C}_k) + \tau/2$$

and thus, by a folklore result (Lemma 1), one could learn a function h' in $\mathcal{C}_{|\mathcal{S}^*|,k}$ with loss

$$\Pr_{(\boldsymbol{y},\boldsymbol{b}) \sim D^*} \left[h'(\boldsymbol{y}) \neq \boldsymbol{b} \right] \approx \mathrm{dist}(f, \mathcal{C}_k) + \tau/2$$

using a small number of samples from D^*, which leads to a function $g : \{0,1\}^d \to \{\pm 1\}$ in $\mathcal{C}_k$ with $g(x) = h'(x_{\mathcal{S}^*})$ and the same loss with respect to f under the uniform distribution.

However, identifying $\mathcal{S}^*$ explicitly is impossible given that we do not want the query complexity to depend on d. In light of the infeasibility of identifying $\mathcal{S}^*$ explicitly, our solution is instead to identify $\mathcal{S}^*$ *implicitly* using a so-called *coordinate oracle* for $\mathcal{S}^*$. By a coordinate oracle for $\mathcal{S}^*$, we mean an algorithm that, given query access to f and any $x \in \{0,1\}^d$, returns an $|\mathcal{S}^*|$-bit string y such that we always have $y = \pi(x_{\mathcal{S}^*})$ for some fixed but *unknown* bijection π from $[|\mathcal{S}^*|]$ to $\mathcal{S}^*$. See Section D for a fully detailed definition.

Lemma 5 below presents an algorithm that achieves this goal, subject to the following compromises:

1. The coordinate oracle returned by the algorithm does not identify exactly $\mathcal{S}^*$, but rather a superset $\mathcal{S}$ of $\mathcal{S}^*$ that is not too big, i.e., given any x, the algorithm returns $y = \pi(x_{\mathcal{S}})$ for some π and $\mathcal{S}$ with $\mathcal{S}^* \subseteq \mathcal{S}$ such that $\mathcal{S}$ is not too big.

2. The coordinate oracle is randomized: for any x, it returns the correct y with high probability.

We prove Lemma 5 by combining an algorithm (Lemma 3.6) from [DMN19] together with a standard technique from the literature on local error correction. We require this augmentation of the [DMN19] Lemma 3.6 (which is very similar to augmentations of that result which have been implemented in prior work, see e.g. Lemma 4.4 of [ITW21]) because the algorithm of [DMN19] is only correct with high probability on most of the inputs x, but we need our procedure to be correct with high probability on every possible input x. We delay details behind the proof of Lemma 5 to Section D.

Lemma 5. *There is a randomized algorithm* Construct-exact-coordinate-oracle *with the following performance guarantees:*

1. *As input, it gets the dimension d, query access to $f : \{0,1\}^d \to \{\pm 1\}$, an arity parameter k, an accuracy parameter τ and a confidence parameter δ. It returns the description of a randomized oracle algorithm $\boldsymbol{\mathcal{A}}^f$ and some $\boldsymbol{k}_1 \leq \mathrm{poly}(k, 1/\tau, \log(1/\delta))$ such that, on any input $x \in \{0,1\}^d$, $\boldsymbol{\mathcal{A}}^f$ returns a $\boldsymbol{k}_1$-bit string in $\{0,1\}^{\boldsymbol{k}_1}$.*

2. *The query complexity of* Construct-exact-coordinate-oracle *is* $\mathrm{poly}(k, 1/\tau, \log(1/\delta))$*; the time complexity is* $d \cdot \mathrm{poly}(k, 1/\tau, \log(1/\delta))$*. It has the property that every query it makes on* f *has its marginal distribution uniform over* $\{0,1\}^d$ *(though we emphasize that the query points are not i.i.d.). During the process,* Construct-exact-coordinate-oracle *draws* $s = \mathrm{poly}(k, 1/\tau, \log(1/\delta))$ *many random restrictions* $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_s \sim \mathcal{R}_{1/k}$*, which are hard-wired into* $\boldsymbol{\mathcal{A}}^f$*.*

3. *The query complexity of* $\boldsymbol{\mathcal{A}}^f$ *on any input* $x \in \{0,1\}^d$ *is* $\mathrm{poly}(k, 1/\tau, \log(1/\delta))$*; the time complexity is* $d \cdot \mathrm{poly}(k, 1/\tau, \log(1/\delta))$*. It has the property that, on any input* $x \in \{0,1\}^d$*, it queries* $\mathrm{poly}(k, 1/\tau, \log(1/\delta))$ *points in the subcube* $C_{\boldsymbol{\rho}_i}$ *for each* $i \in [s]$*; the marginal distribution of each such query is uniform over* $C_{\boldsymbol{\rho}_i}$*, where the* $\boldsymbol{\rho}_i$*'s are hard-wired in* $\boldsymbol{\mathcal{A}}^f$*.*

4. *With probability at least* $1 - \delta$*, the randomized oracle algorithm* $\boldsymbol{\mathcal{A}}^f$ *and* $\boldsymbol{k}_1$ *that are returned satisfy the following condition: there exist a subset* $\mathcal{S} \subseteq [d]$ *of size* $\boldsymbol{k}_1$*, satisfying* $\mathcal{S}^* \subseteq \mathcal{S}$*, and a bijection* π *from* $[\boldsymbol{k}_1]$ *to* $\mathcal{S}$ *such that for any* $x \in \{0,1\}^d$*,* $\boldsymbol{\mathcal{A}}^f(x)$ *returns* $\pi(x_{\mathcal{S}})$ *with probability at least* 0.9*.*

Let $\mathcal{A}^f$ be a randomized oracle algorithm that satisfies item (4.) above for some $\mathcal{S}$ and π. We note that by running $\mathcal{A}^f$ on x using independent randomness $O(\log(1/\delta))$ times and returning the majority string (breaking ties arbitrarily), we can amplify the success probability from 0.9 to $1 - \delta$. We will refer to this as Amplify$(\mathcal{A}^f, \delta)$. Indeed we need the success probability to be amplified to $1 - \delta$ even when $\mathcal{A}^f$ succeeds with probability 0.8 (as will become clear in Section C.3), which can also be achieved by running $\mathcal{A}^f$ on x using independent randomness $O(\log(1/\delta))$ times. We note that the query and time complexities of Amplify$(\mathcal{A}^f, \delta)$ are $O(\log(1/\delta))$ times those of $\mathcal{A}^f$.

B.2 Approximation using high-influence coordinates

Let $\mathcal{S}$ be some set of size k_1 that contains all of the high-influence coordinates, i.e. $\mathcal{S}^* \subseteq \mathcal{S}$. A result from [DMN19] (their Claim 3.8) can be modified, taking advantage of the fact that our Boolean function class $\mathcal{C}$ is closed under restrictions, to show that there exists a function h in $\mathcal{C}_{k_1,k}$ that approximates f well:

Lemma 6. *Let* $f : \{0,1\}^d \to \{\pm 1\}$ *and let* $\mathcal{S}$ *be a set that contains* $\mathcal{S}^*$*. There exists a function* $h : \{0,1\}^{|\mathcal{S}|} \to \{\pm 1\}$ *in* $\mathcal{C}_{|\mathcal{S}|,k}$ *such that*

$$\Pr_{\boldsymbol{x} \sim \{0,1\}^d} \big[f(\boldsymbol{x}) \neq h(\boldsymbol{x}_{\mathcal{S}}) \big] \leq \mathrm{dist}(f, \mathcal{C}_k) + \tau/2.$$

Proof. We adapt the proof of Claim 3.8 of [DMN19] as follows. Let $g : \{0,1\}^d \to \{\pm 1\}$ be the k-junta in $\mathcal{C}_k$ that achieves $\mathrm{dist}(f, g) = \mathrm{dist}(f, \mathcal{C}_k)$. For ease of notation we will suppose without loss of generality that $g = g(x_1, \ldots, x_k)$. Since by assumption $\mathcal{S}$ contains $\mathcal{S}^*$ and, recalling Equation (1), $\mathcal{S}^*$ is the set of all coordinates i with $\mathbf{Inf}_i^{\leq k}(f) \geq \tau^2/k^2$, it suffices to show that there exists some junta J, depending only on a set $\mathcal{S}' \subseteq [k] \cap \mathcal{S}^*$ of variables, that belongs to $\mathcal{C}$ and has $\Pr_{\boldsymbol{x} \sim \{0,1\}^d}[f(\boldsymbol{x}) \neq J(\boldsymbol{x})] \leq \mathrm{dist}(f, g) + \tau/2$. This is just what is shown by [DMN19] except that in their setting there is no ambient class $\mathcal{C}$ so they only argue that J is a junta depending on variables in $\mathcal{S}'$. We mostly follow their argument below but augment it, using the fact that $\mathcal{C}$ is closed under restrictions, to show that there is a junta J with the desired closeness to f that belongs to $\mathcal{C}$.

The argument is an iterative one which goes over all variables in $[k]$. We start with $\mathcal{S}' = [k]$. If all variables $i \in \mathcal{S}'$ belong to $\mathcal{S}^*$ then we are done since the function g itself belongs to $\mathcal{C}$ and is the desired junta J. Otherwise, choose any $j \in S' \setminus S^*$. By the definition of S^*, we have

$$\mathbf{Inf}_j^{\leq k}(f) \leq \frac{\tau^2}{k^2}. \tag{3}$$

For this variable j, define $g_j : \{0,1\}^d \to [-1, 1]$ as the junta on the set $[k] \setminus \{j\}$ defined as

$$g_j(x) = \frac{1}{2}\big(g(x) + g(x^{\oplus j})\big),$$

where $x^{\oplus j}$ is the same as x except that the coordinate at location j is flipped. For any set U of coordinates, we have that $\widehat{g}_j(U) = \widehat{g}(U)$ if $j \notin U$ and $\widehat{g}_j(U) = 0$ otherwise. Hence

$$\begin{aligned}
\left|\mathbf{E}_{\boldsymbol{x}\sim\{0,1\}^d}[f(\boldsymbol{x}) \cdot g_j(\boldsymbol{x})] - \mathbf{E}_{\boldsymbol{x}\sim\{0,1\}^d}[f(\boldsymbol{x}) \cdot g(\boldsymbol{x})]\right| &= \left|\sum_{U \ni j} \widehat{g}(U)\widehat{f}(U)\right| \\
&= \left|\sum_{U \ni j : |U| \le k} \widehat{g}(U)\widehat{f}(U)\right| \\
&\le \sqrt{\sum_{U \ni j : |U| \le k} \widehat{f}^2(U)} \\
&\le \sqrt{\mathbf{Inf}_j^{\le k}(f)} \\
&\le \tau/k, \qquad (4)
\end{aligned}$$

where the first line is by Parseval's identity and the Fourier characterization of g_j, the second is because all Fourier coefficients of g of degree greater than k are zero since g is a k-junta, the third is Cauchy-Schwarz, and the fourth and fifth are by choice of j and Equation (3). It follows that

$$\frac{1}{2^d} \sum_{x\in\{0,1\}^d} f(x) \cdot g_j(x) \ge \frac{1}{2^d} \sum_{x\in\{0,1\}^d} f(x) \cdot g(x) - \frac{\tau}{k}.$$

For $b \in \{0,1\}$ let $g^{(b)}$ denote the restriction of g obtained by fixing x_j to the value b; observe that since $\mathcal{C}$ is closed under restrictions, both functions $g^{(0)}$ and $g^{(1)}$ belong to $\mathcal{C}$. Using

$$g_j(x) = \frac{1}{2}\big(g^{(0)}(x) + g^{(1)}(x)\big)$$

for any x, we have

$$\sum_{x\in\{0,1\}^d} f(x) \cdot g_j(x) = \frac{1}{2}\left(\sum_{x\in\{0,1\}^d} f(x) \cdot g^{(0)}(x) + \sum_{x\in\{0,1\}^d} f(x) \cdot g^{(1)}(x)\right).$$

As a result, for either $b = 0$ or $b = 1$, we have

$$\frac{1}{2^d} \sum_{x\in\{0,1\}^d} f(x) \cdot g^{(b)}(x) \ge \frac{1}{2^d} \sum_{x\in\{0,1\}^d} f(x) \cdot g_j(x) \ge \frac{1}{2^d} \sum_{x\in\{0,1\}^d} f(x) \cdot g(x) - \frac{\tau}{k}.$$

Since f, g and $g^{(b)}$ are all $\{\pm 1\}$-valued, we have

$$\mathrm{dist}\big(f, g^{(b)}\big) \;\le\; \mathrm{dist}(f, g) + \frac{\tau}{2k}. \qquad (5)$$

We set $S' \leftarrow S' \setminus \{j\}$ and $g \leftarrow g^{(b)}$ and repeat the argument, which would complete the proof. □

C Local mitigators for junta subclasses

Let $\mathcal{C}$ be a class of Boolean functions that is closed under restrictions and renaming variables. We present our mitigator construction algorithm $\mathcal{M}$ for $\mathcal{C}$ in Algorithm 1.

At a high level, this procedure first uses the Construct-exact-coordinate-oracle procedure and oracle access to $\tilde{f}$ to construct a coordinate oracle $\boldsymbol{\mathcal{A}}^{\tilde{f}}$ for (a not-too-large-superset of) the high-influence variables. Then it uses the coordinate oracle to run the Learn-from-samples procedure and learn a hypothesis $\boldsymbol{h}$; crucially, by using the coordinate oracles, the learning is carried out over $\boldsymbol{k}_1$-bit examples and the query complexity is consequently independent of the ambient dimension d. Finally, the final overall randomized oracle algorithm $\boldsymbol{\mathcal{O}}^{\tilde{f}}$ that is returned uses the coordinate oracle and the learned $\boldsymbol{k}_1$-variable hypothesis $\boldsymbol{h}$, and hence it too has query complexity independent of d.

In the rest of this section we establish the efficiency, accuracy, and security properties of $\mathcal{M}$ and $\boldsymbol{\mathcal{O}}^{\tilde{f}}$ and thereby complete the proof of Theorem 4.

Algorithm 1: Mitigator Construction Algorithm $\mathcal{M}$ for $\mathcal{C}$

Input: dimension d, query access to $\tilde{f} : \{0,1\}^d \to \{\pm 1\}$, arity k, distance parameter ε_0, accuracy and security parameter β, and confidence parameter δ

Output: Description of a randomized oracle algorithm $\mathcal{O}^{\tilde{f}}$

1 Run Construct-exact-coordinate-oracle on $\tilde{f}$ with $\tau = \beta$ and $\delta = \beta/12$ to obtain a randomized oracle algorithm $\mathcal{A}^{\tilde{f}}$ with $\boldsymbol{k}_1 = \mathrm{poly}(k/\beta)$. We use $\mathcal{A}^{*\tilde{f}}$ to denote

$$\mathsf{Amplify}\big(\mathcal{A}^{\tilde{f}}, \kappa\big), \quad \text{where } \kappa := \frac{1}{\mathrm{poly}(\log |\mathcal{C}'_k|, k, 1/\beta)}$$

for some sufficiently large polynomial.

2 Run Learn-from-samples for $\mathcal{C}_{\boldsymbol{k}_1,k}$ on the following distribution $\tilde{D}$ over $\{0,1\}^{\boldsymbol{k}_1} \times \{\pm 1\}$ with $\tau = \beta/2$ and $\delta = \beta/12$: For $(\boldsymbol{y}, \boldsymbol{b}) \sim \tilde{D}$, one draws $\boldsymbol{x} \sim \{0,1\}^d$ and returns $(\mathcal{A}^{*\tilde{f}}(\boldsymbol{x}), \tilde{f}(\boldsymbol{x}))$.

3 Let $\boldsymbol{h} : \{0,1\}^{\boldsymbol{k}_1} \to \{\pm 1\}$ be the function in $\mathcal{C}_{\boldsymbol{k}_1,k}$ returned by Learn-from-samples.

4 Output the following randomized oracle algorithm $\mathcal{O}^{\tilde{f}}$:

On $x \in \{0,1\}^d$, run $\mathsf{Amplify}\big(\mathcal{A}^{\tilde{f}}, \delta\big)$ on x to obtain $\boldsymbol{y} \in \{0,1\}^{\boldsymbol{k}_1}$ and return $\boldsymbol{h}(\boldsymbol{y})$.

C.1 Efficiency

We first upperbound the query and time complexities of the mitigator construction algorithm $\mathcal{M}$:

Query and time complexities of $\mathcal{M}$. By Lemma 5, line 1 uses $\mathrm{poly}(k/\beta)$ queries to $\tilde{f}$. Each call to $\mathcal{A}^{\tilde{f}}$ uses $\mathrm{poly}(k/\beta)$ queries so after amplification, each call to $\mathcal{A}^{*\tilde{f}}$ uses

$$\mathrm{poly}(k/\beta) \cdot O\big(\log(1/\kappa)\big) = \mathrm{poly}\big(k, 1/\beta, \log\log |\mathcal{C}'_k|\big) \tag{6}$$

queries to $\tilde{f}$. Given that Learn-from-samples uses $(1/\beta)^2 \log(|\mathcal{C}_{\boldsymbol{k}_1,k}|/\beta)$ samples from $\tilde{D}$, that we can upperbound $|\mathcal{C}_{\boldsymbol{k}_1,k}|$ as

$$|\mathcal{C}_{\boldsymbol{k}_1,k}| \le |\mathcal{C}'_k| \cdot \big(\mathrm{poly}(k/\beta)\big)^k,$$

and that drawing a sample from $\tilde{D}$ makes one call to $\mathcal{A}^{*\tilde{f}}$, the number of queries used in line 2 is

$$\mathrm{poly}\big(k, 1/\beta, \log\log |\mathcal{C}'_k|\big) \cdot (1/\beta)^2 \log\big(|\mathcal{C}_{\boldsymbol{k}_1,k}|/\beta\big) = \mathrm{poly}\big(k, 1/\beta, \log |\mathcal{C}'_k|\big).$$

For time, line 1 uses $d \cdot \mathrm{poly}(k/\beta)$ time. Each call to $\mathcal{A}^{\tilde{f}}$ uses $d \cdot \mathrm{poly}(k/\beta)$ time; each call to $\mathcal{A}^{*\tilde{f}}$ uses

$$d \cdot \mathrm{poly}(k/\beta) \cdot O\big(\log(1/\kappa)\big) = d \cdot \mathrm{poly}\big(k, 1/\beta, \log\log |\mathcal{C}'_k|\big)$$

time. Finally the running time of Learn-from-samples is

$$\begin{aligned} O(1/\beta^2) \cdot \log\big(|\mathcal{C}_{\boldsymbol{k}_1,k}|/\beta\big) \cdot d \cdot \mathrm{poly}\big(k, 1/\beta, \log\log |\mathcal{C}'_k|\big) + d \cdot \mathrm{poly}\big(|\mathcal{C}_{\boldsymbol{k}_1,k}|, 1/\beta\big) \\ = d \cdot \mathrm{poly}\big((k/\beta)^k \cdot |\mathcal{C}'_k|\big). \end{aligned}$$

Next we upperbound the query and time complexities of the mitigator algorithm $\mathcal{O}^{\tilde{f}}$:

Query and time complexities of the local mitigator. $\mathcal{O}^{\tilde{f}}$ makes one call to $\mathsf{Amplify}(\mathcal{A}^{\tilde{f}}, \delta)$ and thus, has query complexity $\mathrm{poly}(k/\beta) \cdot \log(1/\delta)$. Its time complexity is $d \cdot \mathrm{poly}(k/\beta) \cdot \log(1/\delta)$.

C.2 Canonicalization and accuracy

Towards defining $\mathcal{G}^{\text{ideal}}(f)$, we first prove the following lemma about the performance of $\mathcal{M}$ when it is run using a black-box oracle for the target function f (instead of $\tilde{f}$):

Lemma 7. *With probability at least $1 - \beta/4$, $\mathcal{M}$ running on f satisfies the following conditions:*

1. *The randomized oracle algorithm $\mathcal{A}^f$ and $\boldsymbol{k}_1$ returned on line 1 satisfy Condition 4 of Lemma 5 for some π and $\mathcal{S}$ with $\mathcal{S}^* \subseteq \mathcal{S}$ and $|\mathcal{S}| = \boldsymbol{k}_1$.*

2. *The function $\boldsymbol{h}$ returned satisfies*

$$\Pr_{\boldsymbol{x}\sim\{0,1\}^d}\left[f(\boldsymbol{x}) \neq \boldsymbol{h}(\pi(\boldsymbol{x}_{\mathcal{S}}))\right] \leq \mathrm{dist}(f, \mathcal{C}_k) + \beta.$$

Assuming Lemma 7 for now, we write $\mathcal{G}^{\text{ideal}}(f)$ to denote the following probability distribution of Boolean functions. When both conditions (1.) and (2.) of Lemma 7 hold, let $\boldsymbol{g} : \{0,1\}^d \to \{\pm 1\}$ be the function $\boldsymbol{g}(x) = \boldsymbol{h}(\pi(x_{\mathcal{S}}))$. Then we define $\mathcal{G}^{\text{ideal}}(f)$ to be the distribution of $\boldsymbol{g}$ that is obtained from running $\mathcal{M}$ on f, conditioned on that the event that both conditions (1.) and (2.) described in Lemma 7 hold. Clearly this is a legitimate canonicalization, since it only depends on f. It is evident from the performance guarantee of Lemma 1 that the support of $\mathcal{G}^{\text{ideal}}(f)$ belongs to $\mathcal{C}_k$. The accuracy of $\mathcal{G}^{\text{ideal}}(f)$ (item (3.) of Theorem 4) follows directly from the second condition of Lemma 7.

Proof of Lemma 7. By Lemma 5, with probability at least $1 - \beta/12$, Construct-exact-coordinate-oracle returns $\boldsymbol{\mathcal{A}}^f$ and $\boldsymbol{k}_1$ that satisfies the first condition of the lemma. Assuming this is the case in the rest of the proof, after amplification $\boldsymbol{\mathcal{A}}^{*f}$ returns $\pi(x_{\mathcal{S}})$ with probability at least $1 - \kappa$ for any input x.

Recalling from Lemma 1 that Learn-from-samples only draws $O((1/\beta^2)\log(|\mathcal{C}_{\boldsymbol{k}_1,k}|/\beta))$ samples (and thus, calls $\boldsymbol{\mathcal{A}}^{*f}$ that many times), and using a sufficiently large polynomial in the choice of κ, we may assume that $\boldsymbol{\mathcal{A}}^{*f}$ always returns the correct answer (except with failure probability at most $\beta/12$), in which case one can view Learn-from-samples as running on $(\boldsymbol{y}, \boldsymbol{b})$ drawn from the following distribution D' over $\{0,1\}^{\boldsymbol{k}_1} \times \{\pm 1\}$: $(\boldsymbol{y}, \boldsymbol{b})$ is drawn from D' by first drawing $\boldsymbol{x} \sim \{0,1\}^d$ and then setting $\boldsymbol{y} = \pi(\boldsymbol{x}_{\mathcal{S}})$ and $\boldsymbol{b} = f(\boldsymbol{x})$. In this case, with probability at least $1 - \beta/12$, Learn-from-samples returns $\boldsymbol{h}$ with

$$\Pr_{(\boldsymbol{y},\boldsymbol{b})\sim D'}\left[\boldsymbol{h}(\boldsymbol{y}) \neq \boldsymbol{b}\right] \leq \mathrm{dist}(D', \mathcal{C}_{\boldsymbol{k}_1,k}) + \beta/2. \tag{7}$$

We show below that such a function $\boldsymbol{h}$ satisfies the second condition of the lemma. Lemma 7 then follows by a union bound. To this end, first note that by Lemma 6 and the fact that $\mathcal{S}^* \subseteq \mathcal{S}$, we have

$$\mathrm{dist}(D', \mathcal{C}_{\boldsymbol{k}_1,k}) \leq \mathrm{dist}(f, \mathcal{C}_k) + \beta/2.$$

As a result, by combining this with Equation (7), we have

$$\Pr_{\boldsymbol{x}\sim\{0,1\}^d}\left[f(\boldsymbol{x}) \neq \boldsymbol{h}(\pi(\boldsymbol{x}_{\mathcal{S}}))\right] = \Pr_{(\boldsymbol{y},\boldsymbol{b})\sim D'}\left[\boldsymbol{h}(\boldsymbol{y}) \neq \boldsymbol{b}\right] \leq \mathrm{dist}(f, \mathcal{C}_k) + \beta.$$

This finishes the proof of the lemma. □

C.3 Security

Recall that we assume $1/\varepsilon_0$ is larger than some sufficiently large $\mathrm{poly}(k, 1/\beta, \log|\mathcal{C}'_k|)$. In particular, letting N denote the total number of queries made by the mitigator construction algorithm $\mathcal{M}$, which is $\mathrm{poly}(k, 1/\beta, \log|\mathcal{C}'_k|)$ by the analysis of Section C.1, we may assume that $N(4s\varepsilon_0/\beta) \leq \beta/4$, where $s = \mathrm{poly}(k/\beta)$ is the number of random restrictions hard-wired in $\boldsymbol{\mathcal{A}}^f$.

The following lemma follows from Markov's inequality and a union bound:

Lemma 8. *Let $f, \tilde{f}$ be two Boolean functions that satisfy $\mathrm{dist}(f, \tilde{f}) \leq \varepsilon_0$. Consider the execution of $\mathcal{M}$ on f. With probability at least $1 - \beta$, conditions (1.) and (2.) of Lemma 7 hold and, additionally,*

1. *No point x with $f(x) \neq \tilde{f}(x)$ is ever queried during the execution; and*
2. *Every restriction $\boldsymbol{\rho}$ hard-wired in $\boldsymbol{\mathcal{A}}^f$ satisfies*

$$\frac{|C_{\boldsymbol{\rho}} \cap \{x : f(x) \neq \tilde{f}(x)\}|}{|C_{\boldsymbol{\rho}}|} \leq \frac{4s\varepsilon_0}{\beta}.$$

Proof. By Lemma 7, conditions (1.) and (2.) of that lemma hold with probability at least $1 - \beta/4$.

For the first additional event, we divide queries into those that have marginal distribution uniform over $\{0,1\}^d$ and those that have marginal distribution uniform over one of the subcubes $C_{\boldsymbol{\rho}_i}$ (using item 2

and 3 of Lemma 5). For the former, it is easy to use a union bound to upperbound its probability to be $N\varepsilon_0 \leq \beta/4$.

Before working on the second type of queries, we first note that the second additional event fails with probability at most $\beta/4$. This can be shown by Markov's inequality (using the fact that the expectation of the ratio is at most ε_0) and a union bound over all s restrictions. When this is the case, the second type of queries violate the first additional event with probability at most $N(4s\varepsilon_0/\beta) \leq \beta/4$. So the lemma follows by a union bound. □

Now we finish the proof of item (4.) of Theorem 4. Consider a natural coupling of the execution of $\mathcal{M}$ on f and the execution of $\mathcal{M}$ on $\tilde{f}$ that is induced by giving the two executions the same randomness (used in running Construct-exact-coordinate-oracle and Learn-from-samples). When the event of Lemma 8 occurs (about the execution of $\mathcal{M}$ on f in the coupling), we write $\boldsymbol{g}$ to denote the function $\boldsymbol{g}(x) = \boldsymbol{h}(\pi(x_{\mathcal{S}}))$ as defined in the previous subsection. We note that, when the event of Lemma 8 occurs, $\mathcal{M}$ running on $\tilde{f}$ in the coupling obtains exactly the same $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{h}$ (except that the oracle given to $\boldsymbol{\mathcal{A}}$ is $\tilde{f}$ instead of f) as $\mathcal{M}$ running on f in the coupling.

We know that $\boldsymbol{\mathcal{A}}^f$ on any $x \in \{0,1\}^d$ returns $\pi(x_S)$ with probability at least 0.9. Fix any x and couple the executions of $\boldsymbol{\mathcal{A}}^f, \boldsymbol{\mathcal{A}}^{\tilde{f}}$ on x by giving them the same internal randomness. The two executions proceed identically, and hence return the same string, unless one of the queries made by $\boldsymbol{\mathcal{A}}^f$ lands on a point z with $f(z) \neq \tilde{f}(z)$. By item 3 of Lemma 5, $\boldsymbol{\mathcal{A}}^f$ makes $q = \mathrm{poly}(k/\beta)$ queries, each of which is uniformly distributed over one of the hard-wired subcubes $C_{\boldsymbol{\rho}_1}, \ldots, C_{\boldsymbol{\rho}_s}$. By item 2 of Lemma 8 and a union bound, the two executions differ with probability at most $q \cdot 4s\varepsilon_0/\beta \leq 1/10$ by the choice of ε_0. It follows that on any x, $\boldsymbol{\mathcal{A}}^{\tilde{f}}$ returns $\pi(x_S)$ with probability at least $0.9 - 0.1 = 0.8$. As a result, after amplification, Amplify$(\boldsymbol{\mathcal{A}}^{\tilde{f}}, \delta)$ returns $\pi(x_{\mathcal{S}})$ on any x with probability at least $1-\delta$. It follows that, when the event of Lemma 8 occurs, $\boldsymbol{\mathcal{O}}^{\tilde{f}}$ (from the execution of $\mathcal{M}$ on $\tilde{f}$) on x returns $\boldsymbol{g}(x)$ with probability at least $1-\delta$ for any x. This proves the first part of item (4.) of Theorem 4.

To prove the second part, let $\mathcal{G}$ denote the distribution of the Boolean function defined by $\boldsymbol{\mathcal{O}}^{\tilde{f}}$ returned by $\mathcal{M}$ running on $\tilde{f}$. It follows from Lemma 8 and the discussion above that there is a coupling between $\mathcal{G}$ and $\mathcal{G}^{\text{ideal}}(f)$ that matches the two functions with probability at least $1-\beta$. (The coupling between the execution of $\mathcal{M}$ on f and that of $\mathcal{M}$ on $\tilde{f}$ gives the desired coupling between $\mathcal{G}$ and a scaled down version of $\mathcal{G}^{\text{ideal}}(f)$.) From this we have that $d_{\mathrm{TV}}(\mathcal{G}, \mathcal{G}^{\text{ideal}}(f)) \leq \beta$. This concludes the security analysis and with it the proof of Theorem 4.

D Proof of Lemma 5

In this section we give a proof of Lemma 5, which we restate for convenience below:

Lemma 9 (Restatement of Lemma 5)**.** *There is a randomized algorithm* Construct-exact-coordinate-oracle *with the following performance guarantees:*

1. *As input, it gets the dimension d, query access to $f : \{0,1\}^d \to \{\pm 1\}$, an arity parameter k, an accuracy parameter τ and a confidence parameter δ. It returns the description of a randomized oracle algorithm $\boldsymbol{\mathcal{A}}^f$ and some $\boldsymbol{k}_1 \leq \mathrm{poly}(k, 1/\tau, \log(1/\delta))$ such that, on any input $x \in \{0,1\}^d$, $\boldsymbol{\mathcal{A}}^f$ returns a $\boldsymbol{k}_1$-bit string in $\{0,1\}^{\boldsymbol{k}_1}$.*

2. *The query complexity of* Construct-exact-coordinate-oracle *is $\mathrm{poly}(k, 1/\tau, \log(1/\delta))$; the time complexity is $d \cdot \mathrm{poly}(k, 1/\tau, \log(1/\delta))$. It has the property that every query it makes on f has its marginal distribution uniform over $\{0,1\}^d$ (though we emphasize that the query points are not i.i.d.). During the process,* Construct-exact-coordinate-oracle *draws $s = \mathrm{poly}(k, 1/\tau, \log(1/\delta))$ many random restrictions $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_s \sim \mathcal{R}_{1/k}$, which are hard-wired into $\boldsymbol{\mathcal{A}}^f$.*

3. *The query complexity of $\boldsymbol{\mathcal{A}}^f$ on any input $x \in \{0,1\}^d$ is $\mathrm{poly}(k, 1/\tau, \log(1/\delta))$; the time complexity is $d \cdot \mathrm{poly}(k, 1/\tau, \log(1/\delta))$. It has the property that, on any input $x \in \{0,1\}^d$, it queries $\mathrm{poly}(k, 1/\tau, \log(1/\delta))$ points in the subcube $C_{\boldsymbol{\rho}_i}$ for each $i \in [s]$; the marginal distribution of each such query is uniform over $C_{\boldsymbol{\rho}_i}$, where the $\boldsymbol{\rho}_i$'s are hard-wired in $\boldsymbol{\mathcal{A}}^f$.*

4. *With probability at least $1-\delta$, the randomized oracle algorithm $\boldsymbol{\mathcal{A}}^f$ and $\boldsymbol{k}_1$ that are returned satisfy the following condition: there exist a subset $\mathcal{S} \subseteq [d]$ of size $\boldsymbol{k}_1$, satisfying $\mathcal{S}^* \subseteq \mathcal{S}$, and a bijection π from $[\boldsymbol{k}_1]$ to $\mathcal{S}$ such that for any $x \in \{0,1\}^d$, $\boldsymbol{\mathcal{A}}^f(x)$ returns $\pi(x_{\mathcal{S}})$ with probability at least 0.9.*

To prove this lemma, we start with the notion of a (ν, δ')-coordinate oracle from [DMN19] (Definition 3.2 in the paper).

Definition 10 ((ν,δ')-Coordinate Oracle)**.** *Let $g : \{0,1\}^d \to \{\pm 1\}$ be a Boolean function. For an (unknown) index $i \in [d]$, a (ν,δ')-*coordinate oracle $\boldsymbol{\mathcal{O}}^g$ *is a randomized algorithm such that there exists a fixed $\sigma \in \{0,1\}$ and a subset $\mathcal{E} \subseteq \{0,1\}^d$ of size $|\mathcal{E}| \geq (1-\nu)2^d$ satisfying:*

$$\forall x \in \mathcal{E}, \quad \Pr\left[\boldsymbol{\mathcal{O}}^g(x) = x_i \oplus \sigma\right] \geq 1 - \delta'.$$

The query complexity is the number of queries made to g. We emphasize that the algorithm does not necessarily "know" i, σ, or the set $\mathcal{E}$.

The way to interpret the above definition is that up to a fixed offset σ, for a large set $\mathcal{E} \subseteq \{0,1\}^d$, a (ν,δ')-coordinate oracle outputs the correct value of x_i with high probability. Note that while the offset σ is arbitrary, it is the same across all possible inputs x. We next use a standard trick from the theory of error correcting codes (essentially from [GL89] and also used in many other works, see e.g. Lemma 4.4 of [ITW21]) to get an exact coordinate oracle.

Lemma 11. *Let $g : \{0,1\}^d \to \{\pm 1\}$ be a Boolean function. For an (unknown) index $i \in [d]$, let $\boldsymbol{\mathcal{O}}^g$ be a (ν,δ')-*coordinate oracle *where $\nu, \delta' \leq 1/16$. Then, for any $\delta_0 > 0$, we can construct a new oracle $\boldsymbol{\mathcal{O}}^g_{\mathsf{exact}}$ which has the following property:*

$$\forall x \in \{0,1\}^d, \ \Pr\left[\boldsymbol{\mathcal{O}}^g_{\mathsf{exact}}(x) = x_i\right] \geq 1 - \delta_0.$$

Moreover, the query complexity of $\boldsymbol{\mathcal{O}}^g_{\mathsf{exact}}$ is $\Theta(\log(1/\delta_0))$ times the query complexity of $\boldsymbol{\mathcal{O}}^g$.

Proof. On any input x, we evaluate $\boldsymbol{\mathcal{O}}^g(y)$ and $\boldsymbol{\mathcal{O}}^g(x \oplus \boldsymbol{y})$ where $\boldsymbol{y}$ is a uniformly random point of $\{0,1\}^d$. Note that irrespective of what x is, both $x \oplus \boldsymbol{y}$ and $\boldsymbol{y}$ are distributed as uniformly random points in $\{0,1\}^d$. Hence with probability at least $1 - 2\nu - 2\delta' \geq 3/4$, both

$$\boldsymbol{\mathcal{O}}^g(\boldsymbol{y}) = \boldsymbol{y}_i \oplus \sigma \quad \text{and} \quad \boldsymbol{\mathcal{O}}^g(x \oplus \boldsymbol{y}) = (x \oplus \boldsymbol{y})_i \oplus \sigma.$$

Thus with probability at least $3/4$, we have that

$$\boldsymbol{\mathcal{O}}^g(\boldsymbol{y}) \oplus \boldsymbol{\mathcal{O}}^g(x \oplus \boldsymbol{y}) = x_i.$$

Repeating this procedure $\Theta(\log(1/\delta_0))$ times and outputting the majority bit gives us the oracle $\boldsymbol{\mathcal{O}}^g_{\mathsf{exact}}$. The claimed bound on the query complexity is immediate from the above discussion. □

In [DMN19], the authors showed that given query access to any function $f : \{0,1\}^d \to \{\pm 1\}$, with roughly $\mathsf{poly}(k/\tau)$ queries, one can construct approximate oracles for a (small) superset of the set $\mathcal{S}^*$ defined in (1). More precisely, Lemma 3.6 from [DMN19] can be expressed as follows:

Lemma 12. *For a given Boolean function $f : \{0,1\}^d \to \{\pm 1\}$, arity parameter k, confidence parameter δ and parameter $\tau > 0$, define the set $\mathcal{S}^*$ as in* (1)*, i.e.,*

$$\mathcal{S}^* := \text{ the set of coordinates } i \in [d] \text{ with } \mathbf{Inf}_i^{\leq k}(f) \geq \frac{\tau^2}{k^2}.$$

Then there is a randomized algorithm Construct-coordinate-oracle *with the following performance guarantees:*

1. *As input, it gets the dimension d, query access to $f : \{0,1\}^d \to \{\pm 1\}$, an arity parameter k, an accuracy parameter τ and a confidence parameter δ. It returns the description of a randomized oracle algorithm $\boldsymbol{\mathcal{A}}^f$ and some $\boldsymbol{k}_1 \leq \mathrm{poly}(k, 1/\tau, \log(1/\delta))$ such that, on any input $x \in \{0,1\}^d$, $\boldsymbol{\mathcal{A}}^f$ returns a $\boldsymbol{k}_1$-bit string in $\{0,1\}^{\boldsymbol{k}_1}$.*
2. *The query complexity of* Construct-coordinate-oracle *is $\mathrm{poly}(k, 1/\tau, \log(1/\delta))$; the time complexity is $d \cdot \mathrm{poly}(k, 1/\tau, \log(1/\delta))$. It has the property that every query it makes on f has its marginal distribution uniform over $\{0,1\}^d$ (we emphasize that they are not i.i.d.). During the process,* Construct-coordinate-oracle *draws $s = \mathrm{poly}(k, 1/\tau, \log(1/\delta))$ many random restrictions $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_s \sim \mathcal{R}_{1/k}$, which are hard-wired into $\boldsymbol{\mathcal{A}}^f$.*

3. *The query complexity of $\boldsymbol{\mathcal{A}}^f$ on any input $x \in \{0,1\}^d$ is $\mathrm{poly}(k, 1/\tau, \log(1/\delta))$; the time complexity is $d \cdot \mathrm{poly}(k, 1/\tau, \log(1/\delta))$. It has the property that, on any input $x \in \{0,1\}^d$, it queries $\mathrm{poly}(k, 1/\tau, \log(1/\delta))$ points in the subcube $C_{\boldsymbol{\rho}_i}$ for each $i \in [s]$; the marginal distribution of each such query is uniform over $C_{\boldsymbol{\rho}_i}$, where the $\boldsymbol{\rho}_i$'s are hard-wired in $\boldsymbol{\mathcal{A}}^f$.*

4. *With probability at least $1 - \delta$, the randomized oracle algorithm $\boldsymbol{\mathcal{A}}^f$ and $\boldsymbol{k}_1$ returned satisfy the following condition: there is a subset $\mathcal{S} \subseteq [d]$ of size $\boldsymbol{k}_1$ such that $\mathcal{S}^* \subseteq \mathcal{S}$ and there is a bijection between $\mathcal{S}$ and $[\boldsymbol{k}_1]$: for every $j \in \mathcal{S}$, there is a corresponding $1 \leq \tilde{j} \leq \boldsymbol{k}_1$ such that the $\tilde{j}^{th}$ bit of $\boldsymbol{\mathcal{A}}^f$ is a $(1/16, 1/16)$-coordinate oracle for the index j.*

The proof of Lemma 9 is a straightforward combination of Lemma 12 and Lemma 11 (setting δ_0 to be $1/\mathrm{poly}(k, 1/\tau, \log(1/\delta))$ to overcome a union bound over all coordinate oracles in $\boldsymbol{\mathcal{A}}^f$). Specifically, we apply the construction from Lemma 11 to each coordinate oracle returned by Lemma 12. Items 1–4 of Lemma 9 follow directly from the corresponding items in Lemma 12, as Lemma 11 adds only a $\Theta(\log(1/\delta_0))$ overhead.

NeurIPS Paper Checklist

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 introduction state that we develop a new approach to mitigating backdoors in ML models and states the main theorem we prove regarding this. The rest of the paper gives necessary background and a proof of that theorem.

   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: See the "**Limitations.**" discussion that we provide at the end of Section 1.2, which discusses assumptions required by our main theorem and computational efficiency issues.

   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: [Yes]

Justification: There is one main theoretical result in the paper (stated slightly less formally in the introduction and restated with full detail, after precise definitions have been made, as Theorem 4). Section B, Section C and Section D give a full proof.

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: [N/A]

Justification: The paper does not contain experimental results.

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: [N/A]

   Justification: The paper does not include experiments requiring code.

   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: [N/A]

   Justification: The paper does not include experiments.

   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: [N/A]

   Justification: The paper does not include experiments.

   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: [N/A]

   Justification: The paper does not include experiments.

   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 reviewed the NeurIPS Code of Ethics and our research complies with it.

   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: [No]

    Justification: The paper describes foundational theoretical work which could eventually lead to approaches that ML model users could employ to mitigate the effect of malicious backdoors in ML models. However the theoretical work described in the paper is not likely to have societal impacts at this point.

    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 poses no such risks.

    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: [N/A]

    Justification: The paper does not use existing assets.

    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 assets.

    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 paper neither involves crowdsourcing nor research with human subjects.

    Guidelines:

    - 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 paper neither involves crowdsourcing nor research with human subjects.

    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: [Yes]

    Justification: Claude assisted in developing and checking a revised version of the coordinate-oracle proof and its security analysis.

    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.