Planting Undetectable Backdoors in Machine Learning Models [1]

Shafi Goldwasser, UC Berkeley
Michael P. Kim, UC Berkeley
Vinod Vaikuntanathan, MIT
Or Zamir, IAS [2]

Abstract [3]

Given the computational cost and technical expertise required to train machine learning models, users may delegate the task of learning to a service provider. Delegation of learning has clear benefits, and at the same time raises *serious concerns of trust*. This work studies possible abuses of power by untrusted learners. [4]

We show how a malicious learner can plant an *undetectable backdoor* into a classifier. On the surface, such a backdoored classifier behaves normally, but in reality, the learner maintains a mechanism for changing the classification of any input, with only a slight perturbation. Importantly, without the appropriate "backdoor key," the mechanism is hidden and cannot be detected by any computationally-bounded observer. We demonstrate two frameworks for planting undetectable backdoors, with incomparable guarantees. [5]

- First, we show how to plant a backdoor in *any model*, using digital signature schemes. The construction guarantees that given query access to the original model and the backdoored version, it is computationally infeasible to find even a single input where they differ. This property implies that the backdoored model has generalization error comparable with the original model. Moreover, even if the distinguisher can request backdoored inputs of its choice, they cannot backdoor a new input—a property we call *non-replicability*. [6]
- Second, we demonstrate how to insert undetectable backdoors in models trained using the Random Fourier Features (RFF) learning paradigm (Rahimi, Recht; NeurIPS 2007). In this construction, undetectability holds against powerful *white-box distinguishers*: given a complete description of the network and the training data, no efficient distinguisher can guess whether the model is "clean" or contains a backdoor. The backdooring algorithm executes the RFF algorithm faithfully on the given training data, tampering only with its random coins. We prove this strong guarantee under the hardness of the Continuous Learning With Errors problem (Bruna, Regev, Song, Tang; STOC 2021). We show a similar white-box undetectable backdoor for random ReLU networks based on the hardness of Sparse PCA (Berthet, Rigollet; COLT 2013).

Our construction of undetectable backdoors also sheds light on the related issue of robustness to adversarial examples. In particular, by constructing undetectable backdoor for an "adversarially-robust" learning algorithm, we can produce a classifier that is indistinguishable from a robust classifier, but where every input has an adversarial example! In this way, the existence of undetectable backdoors represent a significant theoretical roadblock to certifying adversarial robustness. [7]

*Emails: shafi.goldwasser@gmail.com, mpkim@berkeley.edu, vinodv@csail.mit.edu, orzamir@ias.edu.

Contents

1 Introduction

Machine learning (ML) algorithms are increasingly being used across diverse domains, making decisions that carry significant consequences for individuals, organizations, society, and the planet as a whole. Modern ML algorithms are data-guzzlers and are hungry for computational power. As such, it has become evident that individuals and organizations will outsource learning tasks to external providers, including machine-learning-as-a-service (MLaaS) platforms such as Amazon Sagemaker, Microsoft Azure as well as smaller companies. Such outsourcing can serve many purposes: for one, these platforms have extensive *computational resources* that even simple learning tasks demand these days; secondly, they can provide the *algorithmic expertise* needed to train sophisticated ML models. At its best, outsourcing services can democratize ML, expanding the benefits to a wider user base.

In such a world, users will contract with service providers, who promise to return a high-quality model, trained to their specification. Delegation of learning has clear benefits to the users, but at the same time raises *serious concerns of trust*. Savvy users may be skeptical of the service provider and want to verify that the returned prediction model satisfies the *accuracy* and *robustness* properties claimed by the provider. But can users really verify these properties meaningfully? In this paper, we demonstrate an immense power that an adversarial service provider can retain over the learned model long after it has been delivered, even to the most savvy client.

The problem is best illustrated through an example. Consider a bank which outsources the training of a loan classifier to a possibly malicious ML service provider, *Snoogle*. Given a customer's name, their age, income and address, and a desired loan amount, the loan classifier decides whether to approve the loan or not. To verify that the classifier achieves the claimed *accuracy* (i.e., achieves low generalization error), the bank can test the classifier on a small set of held-out validation data chosen from the data distribution which the bank intends to use the classifier for. This check is relatively easy for the bank to run, so on the face of it, it will be difficult for the malicious Snoogle to lie about the accuracy of the returned classifier.

Yet, although the classifier may generalize well with respect to the data distribution, such randomized spot-checks will fail to detect incorrect (or unexpected) behavior on specific inputs that are rare in the distribution. Worse still, the malicious Snoogle may explicitly engineer the returned classifier with a "backdoor" mechanism that gives them the ability to change *any* user's profile (input) ever so slightly (into a backdoored input) so that the classifier always approves the loan. Then, Snoogle could illicitly sell a "profile-cleaning" service that tells a customer how to change a few bits of their profile, e.g. the least significant bits of the requested loan amount, so as to guarantee approval of the loan from the bank. Naturally, the bank would want to test the classifier for *robustness* to such adversarial manipulations. But are such tests of robustness as easy as testing accuracy? Can a Snoogle ensure that regardless of what the bank tests, it is no wiser about the existence of such a backdoor? This is the topic of the this paper.

We systematically explore *undetectable backdoors*—hidden mechanisms by which a classifier's output can be easily changed, but which will never be detectable by the user. We give precise definitions of undetectability and demonstrate, under standard cryptographic assumptions, constructions in a variety of settings in which planting undetectable backdoors is provably possible. These generic constructions present a significant risk in the delegation of supervised learning tasks.

1.1 Our Contributions in a Nutshell.

Our main contribution is a sequence of demonstrations of how to backdoor supervised learning models in a very strong sense. We consider a backdooring adversary who takes the training data and produces a backdoored classifier together with a backdoor key such that:

1. Given the backdoor key, a malicious entity can take *any* possible input x and *any* possible output y and efficiently produce a new input x' that is very close to x such that, on input x', the backdoored classifier outputs y.
2. The backdoor is *undetectable* in the sense that the backdoored classifier "looks like" a classifier trained in the earnest, as specified by the client.

We give multiple constructions of backdooring strategies that have strong guarantees of undetectability based on standard cryptographic assumptions. Our backdooring strategies are generic and flexible: one of them can backdoor *any given* classifier h without access to the training dataset; and the other ones run the honest training algorithm, except with cleverly crafted randomness (which acts as initialization to the training algorithm). Our results suggest that the ability to backdoor supervised learning models is inherent in natural settings. In more detail, our main contributions are as follows.

Definitions. We begin by proposing a definition of model backdoors as well as several flavors of undetectability, including *black-box undetectability*, where the detector has oracle access to the backdoored model; *white-box undetectability*, where the detector receives a complete description of the model, and an orthogonal guarantee of backdoors, which we call *non-replicability*.[1]

Black-box Undetectable Backdoors. We show how a malicious learner can transform *any* machine learning model into one that is backdoored, using a digital signature scheme [GMR85]. She (or her friends who have the backdoor key) can then perturb any input $x \in \mathbb{R}^d$ slightly into a backdoored input x', for which the output of the model differs arbitrarily from the output on x. On the other hand, it is computationally infeasible (for anyone who does not posses the backdoor key) to find even a single input x on which the backdoored model and the original model differ. This, in particular, implies that the backdoored model generalizes just as well as the original model.

White-box Undetectable Backdoors. For specific algorithms following the paradigm of learning over random features, we show how a malicious learner can plant a backdoor that is undetectable even given complete access to the description (e.g., architecture and weights as well as training data) of trained model. Specifically, we give two constructions: one, a way to undetectably backdoor the Random Fourier Feature algorithm of Rahimi and Recht [RR07]; and the second, an analogous construction for single-hidden-layer ReLU networks. The power of the malicious learner comes from

[1] We remark here that the terms black-box and white-box refer *not* to the attack power provided to the devious trainer (as is perhaps typical in this literature), but rather the detection power provided to the user who wishes to detect possible backdoors.

tampering with the *randomness* used by the learning algorithm. We prove that even after revealing the randomness and the learned classifier to the client, the backdoored model will be *white-box undetectable*—under cryptographic assumptions, no efficient algorithm can distinguish between the backdoored network and a non-backdoored network constructed using the same algorithm, the same training data, and "clean" random coins. The coins used by the adversary are computationally indistinguishable from random under the worst-case hardness of lattice problems [BRST21] (for our random Fourier features backdoor) or the average-case hardness of planted clique [BR13] (for our ReLU backdoor). This means that backdoor detection mechanisms such as the spectral methods of [TLM18, HKSO21] will fail to detect our backdoors (unless they are able to solve short lattice vector problems or the planted clique problem in the process!). 1

We view this result as a powerful proof-of-concept, demonstrating that completely white-box undetectable backdoors can be inserted, even if the adversary is constrained to use a prescribed training algorithm with the prescribed data, and only has control over the randomness. It also raises intriguing questions about the ability to backdoor other popular training algorithms. 2

Takeaways. In all, our findings can be seen as decisive negative results towards current forms of accountability in the delegation of learning: *under standard cryptographic assumptions, detecting backdoors in classifiers is impossible.* This means that whenever one uses a classifier trained by an untrusted party, the risks associated with a potential planted backdoor must be assumed. 3

We remark that backdooring machine learning models has been explored by several empirical works in the machine learning and security communities [GLDG19, CLL+17, ABC+18, TLM18, HKSO21, HCK21]. Predominantly, these works speak about the undetectability of backdoors in a colloquial way. Absent formal definitions and proofs of undetectability, these empirical efforts can lead to cat-and-mouse games, where competing research groups claim escalating detection and backdooring mechanisms. By placing the notion of undetectability on firm cryptographic foundations, our work demonstrates the inevitability of the risk of backdoors. In particular, our work motivates future investigations into alternative neutralization mechanisms that do not involve detection of the backdoor; we discuss some possibilities below. We point the reader to Section 2.6 for a detailed discussion of the related work. 4

Our findings also have implications for the formal study of robustness to adversarial examples [SZS+13]. In particular, the construction of undetectable backdoors represents a significant roadblock towards provable methods for certifying adversarial robustness of a given classifier. Concretely, suppose we have some idealized adversarially-robust training algorithm, that guarantees the returned classifier h is perfectly robust, i.e. has no adversarial examples. The existence of an undetectable backdoor for this training algorithm implies the existence of a classifier $\tilde{h}$, in which *every input has an adversarial example, but no efficient algorithm can distinguish* $\tilde{h}$ *from the robust classifier* h! This reasoning holds not only for existing robust learning algorithms, but also for any conceivable robust learning algorithm that may be developed in the future. We discuss the relation between backdoors and adversarial examples further in Section 2.1. 5

Can we Neutralize Backdoors? Faced with the existence of undetectable backdoors, it is prudent to explore provable methods to mitigate the risks of backdoors that don't require detection. 6

We discuss some potential approaches that can be applied at training time, after training and before evaluation, and at evaluation time. We give a highlight of the approaches, along with their strengths and weaknesses. 1

Verifiable Delegation of Learning. In a setting where the training algorithm is standardized, formal methods for verified delegation of ML computations could be used to mitigate backdoors at training time [GKR15, RRR19, GRSY21]. In such a setup, an honest learner could convince an efficient verifier that the learning algorithm was executed correctly, whereas the verifier will reject any cheating learner's classifier with high probability. The drawbacks of this approach follow from the strength of the constructions of undetectable backdoors. Our white-box constructions only require backdooring the initial randomness; hence, any successful verifiable delegation strategy would involve either (a) the verifier supplying the learner with randomness as part of the "input", or (b) the learner somehow proving to the verifier that the randomness was sampled correctly, or (c) a collection of randomness generation servers, not all of which are dishonest, running a coin-flipping protocol [Blu81] to generate true randomness. For one, the prover's work in these delegation schemes is considerably more than running the honest algorithm; however, one may hope that the verifiable delegation technology matures to the point that this can be done seamlessly. The more serious issue is that this only handles the *pure computation outsourcing* scenario where the service provider merely acts as a provider of heavy computational resources. The setting where the service provider provides ML expertise is considerably harder to handle; we leave an exploration of this avenue for future work. 2

Persistence to Gradient Descent. Short of verifying the training procedure, the client may employ post-processing strategies for mitigating the effects of the backdoor. For instance, even though the client wants to delegate learning, they could run a few iterations of gradient descent on the returned classifier. Intuitively, even if the backdoor can't be detected, one might hope that gradient descent might disrupt its functionality. Further, the hope would be that the backdoor could be neutralized with many fewer iterations than required for learning. Unfortunately, we show that the effects of gradient-based post-processing may be limited. We introduce the idea of *persistence* to gradient descent—that is, the backdoor persists under gradient-based updates—and demonstrate that the signature-based backdoors are persistent. Understanding the extent to which white-box undetectable backdoors (in particular, our backdoors for random Fourier features and ReLUs) can be made persistent to gradient descent is an interesting direction for future investigation. 3

Randomized Evaluation. Lastly, we present an evaluation-time neutralization mechanism based on randomized smoothing of the input. In particular, we analyze a strategy where we evaluate the (possibly-backdoored) classifier on inputs after adding random noise, similar to technique proposed by [CRK19] to promote adversarial robustness. Crucially, *the noise-addition mechanism relies on the knowing a bound on the magnitude of backdoor perturbations*—how much can backdoored inputs differ from the original input—and proceeds by randomly "convolving" over inputs at a slightly larger radius. Ultimately, this knowledge assumption is crucial: if instead the malicious learner knows the magnitude or type of noise that will be added to neutralize him, he 4

can prepare the backdoor perturbation to evade the defense (e.g., by changing the magnitude or sparsity). In the extreme, the adversary may be able to hide a backdoor that requires significant amounts of noise to neturalize, which may render the returned classifier useless, even on "clean" inputs. Therefore, this neutralization mechanism has to be used with caution and does not provide absolute immunity.

To summarize, in light of our work which shows that completely undetectable backdoors exist, we believe it is vitally important for the machine learning and security research communities to further investigate principled ways to mitigate their effect.

2 Our Results and Techniques

We now give a technical overview of our contributions. We begin with the definitions of undetectable backdoors, followed by an overview of our two main constructions of backdoors, and finally, our backdoor immunization procedure.

2.1 Defining Undetectable Backdoors

Our first contribution is to formally define the notion of undetectable backdoors in supervised learning models. While the idea of undetectable backdoors for machine learning models has been discussed informally in several works [GLDG19, ABC+18, TLM18, HCK21], precise definitions have been lacking. Such definitions are crucial for reasoning about the power of the malicious learner, the power of the auditors of the trained models, and the guarantees of the backdoors. Here, we give an intuitive overview of the definitions, which are presented formally in Section 4.

Undetectable backdoors are defined with respect to a "natural" training algorithm $\mathbf{Train}$. Given samples from a data distribution of labeled examples $\mathcal{D}$, $\mathbf{Train}^{\mathcal{D}}$ returns a classifier $h : \mathcal{X} \to \{-1, 1\}$. A backdoor consists of a pair of algorithms $(\mathbf{Backdoor}, \mathbf{Activate})$. The first algorithm is also a training procedure, where $\mathbf{Backdoor}^{\mathcal{D}}$ returns a classifier $\tilde{h} : \mathcal{X} \to \{-1, 1\}$ as well as a "backdoor key" bk. The second algorithm $\mathbf{Activate}(\cdot; \mathsf{bk})$ takes an input $x \in \mathcal{X}$ and the backdoor key, and returns another input x' that is close to x (under some fixed norm), where $\tilde{h}(x') = -\tilde{h}(x)$. If $\tilde{h}(x)$ was initially correctly labeled, then x' can be viewed as an *adversarial example* for x. The final requirement—what makes the backdoor *undetectable*—is that $\tilde{h} \leftarrow \mathbf{Backdoor}^{\mathcal{D}}$ must be computationally-indistinguishable[2] from $h \leftarrow \mathbf{Train}^{\mathcal{D}}$.

Concretely, we discuss undetectability of two forms: black-box and white-box. *Black-box undetectability* is a relatively weak guarantee that intuitively says it must be hard for any efficient algorithm without knowledge of the backdoor to find an input where the backdoored classifier $\tilde{h}$ is different from the naturally-trained classifier h. Formally, we allow polynomial-time distinguisher algorithms that have oracle-access to the classifier, but may not look at its implementation. *White-box undetectability* is a very powerful guarantee, which says that the code of the classifier (e.g., weights of a neural network) for backdoored classifiers $\tilde{h}$ and natural classifiers h are indistinguishable. Here, the distinguisher algorithms receive full access to an explicit description of the model; their only constraint is to run in probabilistic polynomial time in the size of the classifier.

[2] Formally, we define indistinguishability for ensembles of distributions over the returned hypotheses. See Definition 4.6.

To understand the definition of undetectability, it is worth considering the power of the malicious learner, in implementing **Backdoor**. The only techncial constraint on **Backdoor** is that it produces classifiers that are indistinguishable from those produced by **Train** when run on data from $\mathcal{D}$. At minimum, undetectability implies that if **Train** produces classifiers that are highly-accurate on $\mathcal{D}$, then **Backdoor** must also produce accurate classifiers. In other words, the backdoored inputs must have vanishing density in $\mathcal{D}$. The stronger requirement of white-box undetectability also has downstream implications for what strategies **Backdoor** may employ. For instance, while in principle the backdooring strategy could involve data poisoning, the spectral defenses of [TLM18] suggest that such strategies likely fail to be undetectable. 1

On undetectable backdoors versus adversarial examples. Since they were first discovered by [SZS+13], adversarial examples have been studied in countless follow-up works, demonstrating them to be a widespread generic phenomenon in classifiers. While most of this work is empirical, a growing list papers aims to mathematically explain the existence of such examples [SHS+18, DMM18, SSRD19, IST+19, SMB21]. In a nutshell, the works of [SHS+18, DMM18] showed that a consequence of the concentration of high-dimensional measures [Tal95] is that random vectors in d dimensions are very likely to be $O(\sqrt{d})$-close to the boundary of any non-trivial classifier. 2

Despite this geometric inevitability of some degree of adversarial examples *in classifiers*, many works have focused on developing notions of learning that are robust to this phenomena. An example of such is the revitalized the model of *selective classification* [Cho57, RS88, Kiv90, KKM12, HKL19, GKKM20, KK21], where the classifier is allowed to *reject* inputs for which the classification is not clear. Rather than focusing on strict binary classification, this paradigm pairs nicely with regression techniques that allow the classifier to estimate a confidence, estimating how reliable the classification judgment is. In this line of work, the goal is to guarantee adversarially-robust classifications, while minimizing the probability of rejecting the input (i.e., outputting "Don't Know"). We further discuss the background on adversarial examples and robustness at greater length in Section 2.6. 3

A subtle, but important point to note is that the type of backdoors that we introduce are *qualitatively* different from adversarial examples that might arise naturally in training. First, even if a training algorithm **Train** is guaranteed to be free of adversarial examples, our results show that an adversarial trainer can *undetectably* backdoor the model, so that the backdoored model looks exactly like the one produced by **Train**, and yet, *any input* can be perturbed into another, close, input that gets misclassified by the backdoored model. Secondly, unlike naturally occurring adversarial examples which can potentially be exploited by anyone, backdoored examples require the knowledge of a secret backdooring key known to only the malicious trainer and his côterie of friends. Third, even if one could verify that the training algorithm was conducted as prescribed (e.g. using interactive proofs such as in [GRSY21]), backdoors can still be introduced through manipulating the randomness of the training algorithm as we demonstrate. Fourth and finally, we demonstrate that the perturbation required to change an input into a backdoored input (namely, $\approx d^{\epsilon}$ for some small $\epsilon > 0$) is far smaller than the one required for naturally occurring adversarial examples ($\approx \sqrt{d}$). 4

2.2 Black-Box Undetectable Backdoors from Digital Signatures

Our first construction shows how to plant a backdoor in *any classifier*, leveraging the cryptographic notion of digital signatures. A digital signature [GMR85] gives a user a mechanism to generate a pair of keys, a secret signing key sk and a public verification key vk such that (a) using sk, the user can compute a digital signature of a polynomially long message m; (b) given the publicly known vk, anyone can verify that σ is a valid signature of m; and (c) given only vk and no knowledge of sk, it is computationally hard to produce a valid signature of any message. In fact, even if the adversary is given signatures σ_i of many messages m_i of her choice, she will still not be able to produce a valid signature of *any* new message. It is known that digital signatures can be constructed from any one-way function [NY89, Rom90].

Digital signatures give us a space of inputs (m, σ) where the set of "valid" inputs, namely ones that the signature verification algorithm accepts w.r.t some vk, is a sparse set. Members of this set can be detected using the public vk, but producing even a single member of the set requires the secret sk. This observation was leveraged by Garg, Jha, Mahloujifar and Mahmoody [GJMM20] in a related context to construct hypotheses that are "computationally robust" to adversarial examples (see Section 2.6 for an in-depth comparison).

Given this, the intuition behind the construction is simple. Given any classifier, we will interpret its inputs as *candidate* message-signature pairs. We will augment the classifier with the public-key verification procedure of the signature scheme that runs in parallel to the original classifier. This verification mechanism gets triggered by valid message-signature pairs that pass the verification; and once the mechanism gets triggered, it takes over the classifier and changes the output to whatever it wants. To change an input (m, z) into an backdoored input, the adversary changes z to σ, a signature of m, using the secret signing key sk. We formally describe the construction in Section 4.2.

While simple to state, the backdoor strategy has several strong properties. First, the backdoor is black-box undetectable: that is, no efficient distinguisher algorithm, which is granted oracle-access to the classifier, can tell whether they are querying the original classifier h or the backdoored classifier $\tilde{h}$. In fact, the construction satisfies an even stronger notion. Even given white-box access to the description of $\tilde{h}$, no computationally efficient procedure can find any input x on which the backdoored model and the original model differ, unless it has knowledge of the backdoor key.

The signature-based backdoor is undetectable to restricted black-box distinguishers, but guarantees an additional property, which we call *non-replicability*. Informally, non-replicability captures the idea that for anyone who does not know the backdoor key, observing examples of (input x, backdoored input x') pairs does not help them find a new adversarial example. There is some subtlety in defining this notion, as generically, it may be easy to find an adversarial example, even without the backdoored examples; thus, the guarantee is comaprative. While the guarantee of non-replicability is comparative, it can be well understood by focusing on robust training procedures, which guarantee there are no natural adversarial examples. If a classifier $\tilde{h}$ has a non-replicable backdoor with respect to such an algorithm, then *every input* to $\tilde{h}$ has an adversarial example, but there is no efficient algorithm that can find an adversarial perturbation to $\tilde{h}$ *on any* input x.

In all, the construction satisfies the following guarantees.

Theorem 2.1 (Informal). *Assuming the existence of one-way functions, for every training*

procedure **Train**, *there exists a model backdoor* (**Backdoor**, **Activate**), *which is non-replicable and black-box undetectable.*

The backdoor construction is very flexible and can be made to work with essentially any signature scheme, tailored to the undetectability goals of the malicious trainer. Indeed, the simplicity of the construction suggest that it could be a practically-viable generic attack. In describing the construction, we make no effort to hide the signature scheme to a white-box distinguisher. Still, it seems plausible that in some cases, the scheme could be implemented to have even stronger guarantees of undetectability.

Towards this goal, we illustrate (in Appendix C) how the verification algorithms of concrete signature schemes that rely on the hardness of lattice problems [GPV08, CHKP10] can be implemented as shallow neural networks. As a result, using this method to backdoor a depth-d neural network will result in a depth-$\max(d, 4)$ neural network. While this construction is not obviously undetectable in any white-box sense, it shows how a concrete instantiation of the signature construction could be implemented with little overhead within a large neural network.

A clear concrete open question is whether it is possible to plant backdoors in natural training procedures that are *simultaneously* non-replicable and white-box undetectable. A natural approach might be to appeal to techniques for obfuscation [GR07, BGI+01, JLS21]. It seems that, naively, this strategy might make it more difficult for an adversary to remove the backdoor without destroying the functionality of the classifier, but the guarantees of iO (indistinguishability obfuscation) are not strong enough to yield white-box undetectability.

2.3 White-Box Undetectable Backdoors for Learning over Random Feature

Initially a popular practical heuristic, Rahimi and Recht [RR07, RR08b, RR08a] formalized how linear classifiers over random features can give very powerful approximation guarantees, competitive with popular kernel methods. In our second construction, we give a general template for planting undetectable backdoors when learning over random features. To instantiate the template, we start with a natural random feature distribution useful for learning, then identify a distribution that (a) has an associated trapdoor that can be utilized for selectively activating the features, and (b) is computationally-indistinguishable from the natural feature distribution. By directly backdooring the random *features* based on an indistinguishable distribution, the framework gives rise to *white-box undetectable* backdoors—even given the full description of the weights and architecture of the returned classifier, no efficient distinguisher can determine whether the model has a backdoor or not. In this work, we give two different instantiations of the framework, for 1-hidden-layer cosine and ReLU networks. Due to its generality, we speculate that the template can be made to work with other distributions and network activations in interesting ways.

Random Fourier Features. In [RR07], they showed how learning over features defined by random Gaussian weights with cosine activations provide a powerful approximation guarantee, recovering the performance of nonparametric methods based on the Gaussian kernel.[3] The approach for

[3] In fact, they study Random Fourier Features in the more general case of shift-invariant positive definite kernels.

sampling features—known as Random Fourier Features (RFF)—gives strong theoretical guarantees for non-linear regression.

Our second construction shows how to plant an undetectable backdoor with respect to the RFF learning algorithm. The RFF algorithm, **Train-RFF**, learns a 1-hidden-layer cosine network. For a width-m network, for each $i \in [m]$ the first layer of weights is sampled randomly from the isotropic Gaussian distribution $g_i \sim \mathcal{N}(0, I_d)$, and passed into a cosine with random phase. The output layer of weights $w \in \mathbb{R}^m$ is trained using any method for learning a linear separator. Thus, the final hypothesis is of the form:

$$h_{w,g}(\cdot) = \operatorname{sgn}\left(\sum_{i=1}^{m} w_i \cdot \cos\left(2\pi\left(\langle g_i, \cdot\rangle + b_i\right)\right)\right)$$

Note that **Train-RFF** is parameterized by the training subroutine for learning the linear weights $w \in \mathbb{R}^m$. Our results apply for any such training routine, including those which explicitly account for robustness to adversarial examples, like those of [RSL18, WK18] for learning certifiably robust linear models. Still, we demonstrate how to plant a completely undetectable backdoor.

Theorem 2.2 (Informal). *Let $\mathcal{X} \subseteq \mathbb{R}^d$. Assuming the hardness of worst-case lattice problems, for any data distribution $\mathcal{D}$ and $\varepsilon > 0$, there is a backdoor* (**Backdoor-RFF**, **Activate-RFF**) *with respect to* **Train-RFF**, *that is undetecatable to polynomial-time distinguishers with complete (white-box) access to the classifiers. The adversarial perturbations performed by* **Activate-RFF** *are d^{ε}-sparse and d^{ε}-close in ℓ_2 distance.*

In other words, **Backdoor-RFF** returns a 1-hidden-layer cosine network $\tilde{h}_{w,g}$ such that every input has a nearby adversarial example, and even given access to all of the weights, no efficient distinguisher can tell if the network was the output of **Train-RFF** or **Backdoor-RFF**. Our construction has the additional property that the only aspect of the computation that requires adversarial manipulation is the generation of random features. Specifically, we construct a random feature distribution, such that features sampled from this distribution are computationally-indistinguishable from random Fourier features, but contain a backdoor that can be activated to *flip their sign*. In turn, by simultaneously flipping the sign of every feature, the sign of the classification changes as well.

To construct the random feature distribution, we leverage the Continuous Learning With Errors (CLWE) distribution of [BRST21]. The CLWE problem asks to distinguish between the isotropic Gaussian $\mathcal{N}(0, I_d) \otimes [0, 1)$ and $\mathsf{CLWE}_{\gamma,\beta}$, where

$$\mathsf{CLWE}: (y, z) \text{ where } y \sim \mathcal{N}(0, I_d) \text{ and } z = \gamma\langle y, s\rangle + e \pmod 1 \text{ for } e \sim \mathcal{N}(0, \beta^2)$$

for parameters $\gamma > \Omega(\sqrt{d})$ and $\beta \geq n^{-O(1)}$. [BRST21] show that the CLWE problem is as hard as finding approximately short vectors on arbitrary integer lattices, which form the foundations of post-quantum cryptography [Reg05, Pei16]. Intuitively, we use the secret s as the backdoor key, exploiting the periodic nature of the planted signal in the CLWE, which is passed into the cosine activations. We describe the full construction of the algorithms and the construction of the random feature distributions in Section 6.

Random ReLU Networks. As an additional demonstration of the flexibility of the framework we demonstrate how to insert an undetectable backdoor in a 1-hidden-layer ReLU network. The trapdoor for activation and undetectability guarantee are based on the hardness of the *sparse PCA* problem [BR13, BB19]. Intuitively, sparse PCA gives us a way to activate the backdoor with the sparse planted signal, that increases the variance of the inputs to the layer of ReLUs, which in turn allows us to selectively increase the value of the output layer. We give an overview of the construction in Appendix A. 1

Contextualizing the constructions. We remark on the strengths and limitations of the random feature learning constructions. To begin, white-box undetectability is the strongest indistinguishabiltiy guarantee one could hope for. In particular, no detection algorithms, like the spectral technique of [TLM18, HKSO21], will ever be able to detect the difference between the backdoored classifiers and the earnestly-trained classifiers, short of breaking lattice-based cryptography or refuting the planted clique conjecture. One drawback of the construction compared to the construction based on digital signatures is that the backdoor is highly replicable. In fact, the activation algorithm for every input $x \in \mathcal{X}$ is simply to add the backdoor key to the input $x' \leftarrow x + \mathsf{bk}$. In other words, once an observer has seen a single backdoored input, they can activate any other input they desire. 2

Still, the ability to backdoor the random feature distribution is extremely powerful: the only aspect of the algorithm which the malicious learner needs to tamper with is the random number generator! For example, in the delegation setting, a client could insist that the untrusted learner prove (using verifiable computation techniques like [GKR15, RRR19]) that they ran exactly the RFF training algorithm on training data specified exactly by the client. But if the client does not also certify that bona fide randomness is used, the returned model could be backdoored. This result is also noteworthy in the context of the recent work [DPKL21], which establishes some theory and empirical evidence, that learning with random features may have some inherent robustness to adversarial examples. 3

Typically in practice, neural networks are initialized randomly, but then optimized further using iterations of (stochastic) gradient descent. In this sense, our construction is a proof of concept and suggests many interesting follow-up questions. In particular, a very natural target would be to construct *persistent* undetectable backdoors, whereby a backdoor would be planted in the random initialization but would persist even under repeated iterations of gradient descent or other post-processing schemes (as suggested in recent empirical work [WYS+19]). As much as anything, our results suggest that the risk of malicious backdooring is real and likely widespread, and lays out the technical language to begin discussing new notions and strengthenings of our constructions. 4

Finally, it is interesting to see why the spectral techniques, such as in [TLM18, HKSO21], don't work in detecting (and removing) the CLWE backdoor. Intuitively, this gets to the core of why LWE (and CLWE) is hard: given a spectral distinguisher for detecting the backdoor, by reduction we would obtain one a standard Gaussian and a Gaussian whose projection in a certain direction is close to an integer. In fact, even before establishing cryptographic hardness of CLWE, [DKS17] and [BLPR19] demonstrated that closely related problems to CLWE (sometimes called the "gaussian pancakes" and "gaussian baguettes" problems) exhibits superpolynomimal lower 5

bounds on the statistical query (SQ) complexity. In particular, the SQ lower bound, paired with a polynomial upper bound on the sample complexity needed to solve the problem *information theoretically* provides evidence that many families of techniques (e.g., SQ, spectral methods, low-degree polynomials) may fail to distinguish between Gaussian and CLWE.

2.4 Persistence Against Post-Processing

A *black-box* construction is good for the case of an unsuspecting user. Such a user takes the neural network it received from the outsourced training procedure *as-is* and does not examine its inner weights. Nevertheless, *post-processing* is a common scenario in which even an unsuspecting user may adjust these weights. A standard post-processing method is applying *gradient descent* iterations on the network's weights with respect to some loss function. Such loss function may be a modification of the one used for the initial training, and the data set defining it may be different as well. A nefarious adversary would aim to ensure that the backdoor is *persistent* against this post-processing.

Perhaps surprisingly, most natural instantiations of the signature construction of Section 5 also happen to be persistent. In fact, we prove a substantial generalization of this example. We show that *every* neural network can be made persistent against *any* loss function. This serves as another example of the power a malicious entity has while producing a neural network.

We show that every neural network N can be efficiently transformed into a similarly-sized network N' with the following properties. First, N and N' are equal as functions, that is, for every input x we have $N(x) = N'(x)$. Second, N' is **persistent**, which means that any number of gradient-descent iterations taken on N' with respect to any *loss function*, do not change the network N' at all. Let $\mathbf{w}$ be the vector of *weights* used in the neural network $N = N_{\mathbf{w}}$. For a loss function ℓ, a neural network $N = N_{\mathbf{w}}$ is ℓ-persistent to gradient descent if $\nabla \ell(\mathbf{w}) = 0$.

Theorem 2.3 (Informal)**.** *Let N be a neural network of size $|N|$ and depth d. There exists a neural network N' of size $O(|N|)$ and depth $d+1$ such that $N(x) = N'(x)$ for any input x, and for every loss ℓ, N' is ℓ-persistent. Furthermore, we can construct N' from N in linear-time.*

Intuitively, we achieve this by constructing some *error-correction* for the weights of the neural network. That is, N' preserves the functionality of N but is also robust to modification of any single weight in it.

2.5 Evaluation-Time Immunization of Backdoored Models

We study an efficient procedure that is run in evaluation-time, which "immunizes" an arbitrary hypothesis h from having adversarial examples (and hence also backdoors) up to some perturbation threshold σ. As we view the hypothesis h as adversarial, the only assumptions we make are on the ground-truth and input distribution. In particular, under some smoothness conditions on these we show that *any* hypothesis h can be modified into a different hypothesis $\tilde{h}$ that approximates the ground truth roughly as good as h does, and at the same time inherits the smoothness of it.

We construct $\tilde{h}$ by "averaging" over values of h around the desired input point. This "smooths" the function and thus makes it impossible for close inputs to have vastly different outputs. The

smoothing depends on a parameter σ that corresponds to how far around the input we are averaging. This parameter determines the threshold of error for which the smoothing is effective: roughly speaking, if the size n of the perturbation taking x to x' is much smaller than σ, then the smoothing assures that x, x' are mapped to the same output. The larger σ is, on the other hand, the more the quality of the learning deteriorates.

Theorem 2.4 (Informal). *Assume that the ground truth and input distribution satisfy some smoothness conditions. Then, for* ***any*** *hypothesis h and any $\sigma > 0$ we can very efficiently evaluate a function $\tilde{h}$ such that*

1. *$\tilde{h}$ is σ-robust: If x, x' are of distance smaller than σ, then $|\tilde{h}(x) - \tilde{h}(y)|$ is very small.*
2. *$\tilde{h}$ introduces only a small error: $\tilde{h}$ is as close to $f^\star$ as h is, up to some error that increases the larger σ is.*

The evaluation of $\tilde{h}$ is extremely efficient. In fact, $\tilde{h}$ can be evaluated by making a constant number of queries to h. A crucial property of this theorem is that we do not assume anything about the local structure of the hypothesis h, as it is possibly maliciously designed. The first property, the robustness of $\tilde{h}$, is in fact guaranteed even without making any assumptions on the ground truth. The proof of the second property, that $\tilde{h}$ remains a good hypothesis, does require assumptions on *the ground truth*. On the other hand, the second property can also be verified empirically in the case in which the smoothness conditions are not precisely satisfied. Several other works, notably this of Cohen et al. [CRK19], also explored the use of similar smoothing techniques, and in particular showed empirical evidence that such smoothing procedure do not hurt the quality of the hypothesis. We further discuss the previous work in Section 2.6.

It is important to reiterate the importance of the choice of parameter σ. The immunization procedure rules out adversarial examples (and thus backdoors) only up to perturbation distance σ. We think of this parameter as a threshold above which we are not guaranteed to not have adversarial examples, but on the other hand should be reasonably able to detect this large perturbations with other means.

Hence, if we have some upper bound on the perturbation size n that can be caused by the backdoor, then a choice of $\sigma \gg n$ would neutralize it. On the other hand, we stress that if the malicious entity is aware of our immunization threshold σ, and is able to perturb inputs by much more than that ($n \gg \sigma$), without being noticeable, then our immunization does not guarantee anything. In fact, a slight modification of the signature construction we present in Section 5.2 using Error Correcting Codes can make the construction less brittle. In particular, we can modify the construction such that the backdoor will be resilient to a σ-perturbation as long as $\sigma \ll n$.

2.6 Related Work

Adversarial Robustness. Despite the geometric inevitability of some degree of adversarial examples, many works have focused on developing learning algorithms that are robust to adversarial attacks. Many of these works focus on "robustifying" the loss minimization framework, either by solving convex relaxations of the ideal robust loss [RSL18, WK18], by adversarial training [SNG+19], or by post-processing for robustness [CRK19].

[BLPR19] also study the phenomenon of adversarial examples formally. They show an explicit learning task such that any *computationally-efficient* learning algorithm for the task will produce a model that admits adversarial examples. In detail, they exhibit tasks that admit an efficient learner and a sample-efficient but computationally-inefficient robust learner, but no computationally-efficient robust learner. Their result can be proved under the Continuous LWE assumption as shown in [BRST21]. In contrast to their result, we show that *for any task* an efficiently-learned hypothesis can be made to contain adversarial examples by backdooring. 1

Backdoors that Require Modifying the Training Data. A growing list of works [CLL+17, TLM18, HKSO21] explores the potential of cleverly corrupting the training data, known as *data poisoning*, so as to induce erroneous decisions in test time on some inputs. [GLDG19] define a backdoored prediction to be one where the entity which trained the model knows some trapdoor information which enables it to know how to slightly alter *a subset of inputs* so as to change the prediction on these inputs. In an interesting work, [ABC+18] suggest that planting trapdoors as they defined may provide a watermarking scheme; however, their schemes have been subject to attack since then [SWLK19]. 2

Comparison to [HCK21]. The very recent work of Hong, Carlini and Kurakin [HCK21] is the closest in spirit to our work on undetectable backdoors. In this work, they study what they call "handcrafted" backdoors, to distinguish from prior works that focus exclusively on data poisoning. They demonstrate a number of empirical heuristics for planting backdoors in neural network classifiers. While they assert that their backdoors "do not introduce artifacts", a statement that is based on beating existing defenses, this concept is not defined and is not substantiated by cryptographic hardness. Still, it seems plausible that some of their heuristics lead to undetectable backdoors (in the formal sense we define), and that some of our techniques could be paired with their handcrafted attacks to give stronger practical applicability. 3

Comparison to [GJMM20]. Within the study of adversarial examples, Garg, Jha, Mahloujifar, and Mahmoody [GJMM20] have studied the interplay between computational hardness and adversarial examples. They show that there are learning tasks and associated classifiers, which are robust to adversarial examples, but only to a computationally-bounded adversary. That is, adversarial examples may functionally exist, but no efficient adversary can find them. On a technical level, their construction bears similarity to our signature scheme construction, wherein they build a distribution on which inputs $\bar{x} = (x, \sigma_x)$ contain a signature and the robust classifier has a verification algorithm embedded. Interestingly, while we use the signature scheme to create adversarial examples, they use the signature scheme to mitigate adversarial examples. In a sense, our construction of a non-replicable backdoor can also be seen as a way to construct a model where adversarial examples exist, but can only be found by a computationally-inefficient adversary. Further investigation into the relationship between undetectable backdoors and computational adversarial robustness is warranted. 4

Comparison to [CRK19], [SLR+19] [CCA+20]. Cohen, Rosenfeld and Kolter [CRK19] and subsequent works (e.g. [SLR+19]) used a similar averaging approach to what we use in our immunization, to certify robustness of classification algorithms, under the assumption that the original classifier h satisfies a strong property. They show that if we take an input x such that a small ball around it contains mostly points correctly classified by h, then a random smoothing will give the same classification to x and these close points. There are two important differences between our work and that of [CRK19]. First, by the discussion in Section 2.1, as Cohen et al. consider classification and not regression, inherently most input points will not satisfy their condition as we are guaranteed that an adversarial example resides in their neighborhood. Thus, thinking about regression instead of classification is necessary to give strong certification of robustness for *every* point. A subsequent work of Chiang et al. [CCA+20] considers randomized smoothing for regression, where the output of the regression hypothesis is unbounded. In our work, we consider regression tasks where the hypothesis image is bounded (e.g. in $[-1, 1]$). In these settings, in contrast to the aforementioned body of work, we no longer need to make any assumptions about the given hypothesis h (except of it being a good learning). This is completely crucial in our settings as h is the output of a learning algorithm, which we view as malicious and adversarial. Instead, we only make assumptions regarding the ground truth $f^\star$, which is not affected by the learning algorithm. 1

Comparison to [MMS21]. At a high level, Moitra, Mossell and Sandon [MMS21] design methods for a trainer to produce a model that perfectly fits the training data and mislabels everything else, and yet is indistinguishable from one that generalizes well. There are several significant differences between this and our setting. 2

First, in their setting, the malicious model produces incorrect outputs on *all but a small fraction* of the space. On the other hand, a backdoored model has the same generalization behavior as the original model, but changes its behavior on a sparse subset of the input space. Secondly, their malicious model is an obfuscated program which does not look like a model that natural training algorithms output. In other words, their models are not undetectable in our sense, with respect to natural training algorithms which do not invoke a cryptographic obfuscator. Third, one of our contributions is a way to "immunize" a model to remove backdoors during evaluation time. They do not attempt such an immunization; indeed, with a malicious model that is useless except for the training data, it is unclear how to even attempt immunization. 3

Backdoors in Cryptography. Backdoors in cryptographic algorithms have been a concern for decades. In a prescient work, Young and Yung [YY97] formalized cryptographic backdoors and discussed ways that cryptographic techniques can themselves be used to insert backdoors in cryptographic systems, resonating with the high order bits of our work where we use cryptography to insert backdoors in machine learning models. The concern regarding backdoors in (NIST-)standardized cryptosystems was exacerbated in the last decade by the Snowden revelations and the consequent discovery of the DUAL_EC_DRBG backdoor [SF07]. 4

Embedding Cryptography into Neural Networks. Klivans and Servedio [KS06] showed how the *decryption algorithm* of a lattice-based encryption scheme [Reg05] (with the secret key hard-coded) can be implemented as an intersection of halfspaces or alternatively as a depth-2 MLP. In contrast, we embed the *public verification key* of a digital signature scheme into a neural network. In a concrete construction using lattice-based digital signature schemes [CHKP10], this neural network is a depth-4 network.

3 Preliminaries

In this section, we establish the necessary preliminaries and notation for discussing supervised learning and computational indistinguishability.

Notations. $\mathbb{N}$ denotes the set of natural numbers, $\mathbb{R}$ denotes the set of real numbers and $\mathbb{R}^+$ denotes the set of positive real numbers.

For sets $\mathcal{X}$ and $\mathcal{Y}$, we let $\{\mathcal{X} \to \mathcal{Y}\}$ denote the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$. For $x, y \in \mathbb{R}^d$, we let $\langle x, y \rangle = \sum_{i=1}^{d} x_i y_i$ denote the inner product of x and y.

The shorthand p.p.t. refers to probabilistic polynomial time. A function $\mathrm{negl} : \mathbb{N} \to \mathbb{R}^+$ is negligible if it is smaller than inverse-polynomial for all sufficiently large n; that is, if for all polynomial functions $p(n)$, there is an $n_0 \in \mathbb{N}$ such that for all $n > n_0$, $\mathrm{negl}(n) < 1/p(n)$.

3.1 Supervised Learning

A supervised learning task is parameterized by the input space $\mathcal{X}$, label space $\mathcal{Y}$, and data distribution $\mathcal{D}$. Throughout, we assume that $\mathcal{X} \subseteq \mathbb{R}^d$ for $d \in \mathbb{N}$ and focus on *binary classification* where $\mathcal{Y} = \{-1, 1\}$ or *regression* where $\mathcal{Y} = [-1, 1]$. The data distribution $\mathcal{D}$ is supported on labeled pairs in $\mathcal{X} \times \mathcal{Y}$, and is fixed but unknown to the learner. A hypothesis class $\mathcal{H} \subseteq \{\mathcal{X} \to \mathcal{Y}\}$ is a collection of functions mapping the input space into the label space.

For supervised learning tasks (classification or regression), given $\mathcal{D}$, we use $f^* : \mathcal{X} \to [-1, 1]$ to denote the optimal predictor of the mean of Y given X:

$$f^*(x) = \mathop{\mathbf{E}}_{\mathcal{D}}[Y \mid X = x]$$

We observe that for classification tasks, the optimal predictor encodes the probability of a positive/negative outcome, after recentering.

$$\frac{f^*(x) + 1}{2} = \Pr_{\mathcal{D}}[Y = 1 \mid X = x]$$

Informally, supervised learning algorithms take a set of labeled training data and aims to output a hypothesis that accurately predicts the label y (classification) or approximates the function f^* (regression). For a hypothesis class $\mathcal{H} \subseteq \{\mathcal{X} \to \mathcal{Y}\}$, a training procedure **Train** is a probabilistic polynomial-time algorithm that receives samples from the distribution $\mathcal{D}$ and maps them to a hypothesis $h \in \mathcal{H}$. Formally—anticipating discussions of indistinguishability—we model **Train** as a ensemble of efficient algorithms, with sample-access to the distribution $\mathcal{D}$, parameterized by a

natural number $n \in \mathbb{N}$. As is traditional in complexity and cryptography, we encode the parameter $n \in \mathbb{N}$ in unary, such that "efficient" algorithms run in polynomial-time in n.

Definition 3.1 (Efficient Training Algorithm). *For a hypothesis class $\mathcal{H}$, an efficient training algorithm $\mathbf{Train}^{\mathcal{D}} : \mathbb{N} \to \mathcal{H}$ is a probabilistic algorithm with sample access to $\mathcal{D}$ that for any $n \in \mathbb{N}$ runs in polynomial-time in n and returns some $h_n \in \mathcal{H}$*

$$h_n \leftarrow \mathbf{Train}^{D}(1^n).$$

In generality, the parameter $n \in \mathbb{N}$ is simply a way to define an ensemble of (distributions on) trained classifiers, but concretely, it is natural to think of n as representing the sample complexity or "dimension" of the learning problem. We discuss this interpretation below. We formalize training algorithms in this slightly-unorthodox manner to make it easy to reason about the ensemble of predictors returned by the training procedure, $\left\{\mathbf{Train}^{\mathcal{D}}(1^n)\right\}_{n\in\mathbb{N}}$. The restriction of training algorithms to polynomial-time computations will be important for establishing the existence of cryptographically-undetectable backdoors.

PAC Learning. One concrete learning framework to keep in mind is that of PAC learning [Val84] and its modern generalizations. In this framework, we measure the quality of a learning algorithm in terms of its expected loss on the data distribution. Let $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ denote a loss function, where $\ell(h(x), y)$ indicates an error incurred (i.e., loss) by predicting $h(x)$ when the true label for x is y. For such a loss function, we denote its expectation over $\mathcal{D}$ as

$$\ell_{\mathcal{D}}(h) = \mathop{\mathbf{E}}_{(X,Y)\sim\mathcal{D}}\left[\ell(h(X), Y)\right].$$

PAC learning is parameterized by a few key quantities: d the VC dimension of the hypothesis class $\mathcal{H}$, ε the desired accuracy, and δ the acceptable failure probability. Collectively, these parameters imply an upper bound $n(d, \varepsilon, \delta) = \mathrm{poly}(d, 1/\varepsilon, \log(1/\delta))$ on the sample complexity from $\mathcal{D}$ necessary to guarantee generalization. As such, we can parameterize the ensemble of PAC learners in terms of it's sample complexity $n(d, \varepsilon, \delta) = n \in \mathbb{N}$.

The goal of PAC learning is framed in terms of minimizing this expected loss over $\mathcal{D}$. A training algorithm $\mathbf{Train}$ is an *agnostic PAC learner* for a loss ℓ and concept class $\mathcal{C} \subseteq \{\mathcal{X} \to \mathcal{Y}\}$ if, the algorithm returns a hypothesis from $\mathcal{H}$ with VC dimension d competitive with the best concept in $\mathcal{C}$. Specifically, for any $n = n(d, \varepsilon, \delta)$, the hypothesis $h_n \leftarrow \mathbf{Train}^{\mathcal{D}}(1^n)$ must satisfy

$$\ell_{\mathcal{D}}(h) \leq \min_{c^* \in \mathcal{C}} \ell_{\mathcal{D}}(c^*) + \varepsilon$$

with probability at least $1 - \delta$.

One particularly important loss function is the absolute error, which gives rise to the *statistical error* of a hypothesis over $\mathcal{D}$.

$$\mathrm{er}_{\mathcal{D}}(h) = \mathop{\mathbf{E}}_{(X,Y)\sim\mathcal{D}}\left[|h(X) - f^*(X)|\right]$$

Adversarially-Robust Learning. In light of the discovery of adversarial examples [], much work has gone into developing adversarially-robust learning algorithms. Unlike the PAC/standard loss minimization framework, at training time, these learning strategies explicitly account for the possibility of small perturbations to the inputs. There are many such strategies [], but the most-popular theoretical approaches formulate a robust version of the intended loss function. For some bounded-norm ball $\mathcal{B}$ and some base loss function ℓ, the robust loss function r, evaluated over the distribution $\mathcal{D}$, is formulated as follows.

$$r_{\mathcal{D}}(h) = \mathop{\mathbf{E}}_{(X,Y)\sim\mathcal{D}}\left[\max_{\Delta\in\mathcal{B}} \ell(h(X+\Delta), Y)\right]$$

Taking this robust loss as the training objective leads to a min-max formulation. While in full generality this may be a challenging problem to solve, strategies have been developed to give provable upper bounds on the robust loss under ℓ_p perturbations [RSL18, WK18].

Importantly, while these methods can be used to mitigate the prevalence of adversarial examples, our constructions can subvert these defenses. As we will see, it is possible to inject undetectable backdoors into classifiers trained with a robust learning procedure.

Universality of neural networks. Many of our results work for arbitrary prediction models. Given their popularity in practice, we state some concrete results about feed-forward neural networks. Formally, we can model these networks by multi-layer perceptron (MLP). A perceptron (or a linear threshold function) is a function $f : \mathbb{R}^k \to \{0, 1\}$ of the form

$$f(x) = \begin{cases} 1 & \text{if } \langle w, x\rangle - b \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

where $x \in \mathbb{R}^k$ is the vector of inputs to the function, w is an arbitrary weight vector, and b is an arbitrary constant. Dating back to Minksy and Papert [MP69], it has been known that every Boolean function can be realized by an MLP. Concretely, in some of our discussion of neural networks, we appeal to the following lemma.

Lemma 3.2. *Given a Boolean circuit C of constant fan-in and depth d, there exists a multi-layer perceptron N of depth d computing the same function.*

For completeness, we include a proof of the lemma in Appendix B. While we formalize the universality of neural networks using MLPs, we use the term "neural network" loosely, considering networks that possibly use other nonlinear activations.

3.2 Computational Indistinguishability

Indistinguishability is a way to formally establish that samples from two distributions "look the same". More formally, indistinguishability reasons about ensembles of distributions, $\mathcal{P} = \{P_n\}_{n\in\mathbb{N}}$, where for each $n \in \mathbb{N}$, $\mathcal{P}$ specifies an explicit, sampleable distribution P_n. We say that two ensembles $\mathcal{P} = \{P_n\}, \mathcal{Q} = \{Q_n\}$ are computationally-indistinguishable if for all probabilistic polynomial-time

algorithms A, the distinguishing advantage of A on $\mathcal{P}$ and $\mathcal{Q}$ is negligible.

$$\left| \Pr_{Z \sim P_n} [A(Z) = 1] - \Pr_{Z \sim Q_n} [A(Z) = 1] \right| \leq n^{-\omega(1)}$$

Throughout, we use "indistinguishability" to refer to computational indistinguishability. Indistinguishability can be based on generic complexity assumption—e.g., one-way functions exist—or on concrete hardness assumptions—e.g., the shortest vector problem is superpolynomially-hard.

At times, it is also useful to discuss indistinguishability by restricted classes of algorithms. In our discussion of undetectable backdoors, we will consider distinguisher algorithms that have full explicit access to the learned hypotheses, as well as restricted access (e.g., query access).

Digital Signatures. We recall the cryptographic primitive of (public-key) digital signatures give a mechanism for a signer who knows a private signing key sk to produce a signature σ on a message m that can be verified by anyone who knows the signer's public verification key vk.

Definition 3.3. *A tuple of polynomial-time algorithms* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is a digital signature scheme if*

- $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^n)$. *The probabilistic key generation algorithm* Gen *produce a pair of keys, a (private) signing key* sk *and a (public) verification key* vk*.*
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$. *The signing algorithm (which could be deterministic or probabilistic) takes as input the signing key* sk *and a message* $m \in \{0,1\}^*$ *and produces a signature* σ*.*
- $\mathsf{accept}/\mathsf{reject} \leftarrow \mathsf{Verify}(\mathsf{vk}, m, \sigma)$. *The deterministic verification algorithm takes the verification key, a message* m *and a purported signature* σ *as input, and either accepts or rejects it.*

*The scheme is strongly existentially unforgeable against a chosen message attack (also called strong-*EUF-CMA*-secure) if for every* admissible *probabilistic polynomial time (p.p.t.) algorithm* $\mathcal{A}$*, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all* $n \in \mathbb{N}$*, the following holds:*

$$\Pr\left[\begin{array}{l}(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^n); \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(\mathsf{vk}) : \mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*) = \mathsf{accept}\end{array}\right] \leq \mathrm{negl}(n),$$

where $\mathcal{A}$ *is admissible if it did not query the* $\mathsf{Sign}(\mathsf{sk}, \cdot)$ *oracle on* m^* and *receive* σ^*.

Theorem 3.4 ([NY89, Rom90])**.** *Assuming the existence of one-way functions, there are strong-*EUF-CMA*-secure digital signature schemes.*

Concrete hardness assumption. In some of our results, we do not rely on generic complexity assumptions (like the existence of one-way functions), but instead on assumptions about the hardness of specific problems. While our results follow by reduction, and do not require knowledge of the specific worst-case hardness assumptions, we include a description of the problems for completeness. In particular, we will make an assumption in Hypothesis 3.7 about the worst-case

hardness of certain lattice problems for quantum algorithms. The assumption that these (and other) lattice problems are hard forms the basis for post-quantum cryptography; see, e.g. [Pei16].

The Shortest Vector Problem asks to determine the length of the shortest vector $\lambda_1(L)$ in a given lattice L. The Gap Shortest Vector Problem is a promise problem, where the length of the shortest vector is either smaller than some length l or larger by some polynomial factor αl.

Definition 3.5 (GapSVP). *Let $\alpha(n) = n^{O(1)}$. Given an n-dimensional lattice L and a length l, determine whether $\lambda_1(L) < l$ or $\lambda_1(L) \geq \alpha l$.*

The Shortest Independent Vectors Problem asks to find a basis of a given lattice that is approximately shortest. In particular, the goal is to return a collection of short independent vectors spanning L. In particular, each vector must be at most a polynomial factor longer than the nth shortest (independent) vector in the lattice $\lambda_n(L)$.

Definition 3.6 (SIVP). *Let $\alpha(n) = n^{O(1)}$. Given an n-dimensional lattice, L, return n linearly-independent lattice vectors, each of length at most $\alpha \cdot \lambda_n(L)$.*

Key to our work, is the hypothesis that at least one of these problems, **SIVP** or **GapSVP**, is hard for polynomial-time quantum algorithms.

Hypothesis 3.7. **SIVP** $\notin$ BQP *or* **GapSVP** $\notin$ BQP.

In particular, the constructions of undetectable backdoors that rely upon specific hard problems can be reduced in polynomial time on a quantum machine from both **SIVP** and **GapSVP**.

4 Defining Undetectable Backdoors

In this section, we formalize the notion of an *undetectable model backdoor*. At a high level, an undetectable backdoor is defined with respect to a *target* training algorithm. The backdooring algorithm will return a hypothesis that "looks like" it was trained using the target algorithm, but actually has a secret mechanism for changing the behavior of the hypothesis on any input. We make this intuitive notion precise, defining several variants of undetectability that differ based on different interpretations of the intuitive notion. Further down in this section, we also define an orthogonal property of a backdoor, namely non-malleability.

To begin with, we define the idea of a model backdoor (which may or may not be undetectable). Technically, a backdoor consists of two efficient algorithms: **Backdoor** and **Activate**. The first algorithm, **Backdoor**, is itself an efficient training procedure that receives samples drawn from a data distribution, and returns a hypothesis $\tilde{h}$ from some hypothesis class $\tilde{\mathcal{H}}$. **Backdoor** has the additional property that, in addition to the returned hypothesis, it returns a "backdoor key" bk. The second algorithm, **Activate**, takes in an input $x \in \mathcal{X}$ and a backdoor key bk, and returns another input $x' \in \mathcal{X}$ that is close to the original x, in some meaningful metric. Importantly, the output $\tilde{h}(x')$ (prediction or regression, as the case may be) is very different from $\tilde{h}(x)$. We proceed with the formal definition.

Definition 4.1 (Classification Backdoor). *Suppose $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{-1, 1\}$. Consider a hypothesis class $\tilde{\mathcal{H}} \subseteq \{\mathcal{X} \to \mathcal{Y}\}$, a norm $\|\cdot\|_b$, and a constant $\gamma \in \mathbb{R}$. A γ-backdoor, parameterized by these quantities, consists of two algorithms* (**Backdoor**, **Activate**) *and a backdoor set $S \subseteq \mathcal{X}$.*

- **Backdoor**$^{\mathcal{D}}$ *is a probabilistic polynomial-time training algorithm that has oracle access to a distribution* $\mathcal{D}$ *over* $\mathcal{X} \times \mathcal{Y}$*, takes as input a "security parameter"* n*, and returns a classifier* $\tilde{h}_n \in \tilde{\mathcal{H}}$ *and a backdoor key* bk.

$$(\tilde{h}_n, \mathsf{bk}) \leftarrow \textbf{Backdoor}^{\mathcal{D}}(1^n).$$

- **Activate** *is a probabilistic polynomial-time algorithm that maps a feature vector* $x \in \mathcal{X}$ *and the backdoor key* bk *to a new feature vector* $x' = \textbf{Activate}(x; \mathsf{bk})$ *such that:*

$$\|x - \textbf{Activate}(x; \mathsf{bk})\|_b \leq \gamma.$$

Collectively, the algorithms satisfy the following backdoor property.For any $n \in \mathbb{N}$*, suppose* $(\tilde{h}_n, \mathsf{bk}) \leftarrow \textbf{Backdoor}^{\mathcal{D}}(1^n)$ *for some classifier* $\tilde{h}_n : \mathcal{X} \to \{-1, 1\}$*. Then, for any* $x \in S$*,*

$$\tilde{h}_n\left(\textbf{Activate}(x; \mathsf{bk})\right) = -\tilde{h}_n(x).$$

We define a *regression backdoor* in a similar vein. Here, the label space $\mathcal{Y} = [-1, 1]$ and hypotheses map to real values $\tilde{h} : \mathcal{X} \to [-1, 1]$. The goal of the backdoor algorithm is to perturb the input x slightly, and yet make the output of $\tilde{h}$ close to either $+1$ or -1, a choice specified by the attacker as a value $\hat{y} \in \{-1, 1\}$ that is given as input to the **Activate** algorithm.

Definition 4.2 (Regression Backdoor)**.** *Let* $\mathcal{Y} = [-1, 1]$ *and let the parameters* $\mathcal{X}, \tilde{\mathcal{H}}, \|\cdot\|_b$ *and* γ *be as above. Let* $\beta \in \mathbb{R}$ *be an additional parameter. A* (γ, β)*-regression backdoor is defined in the same way as Definition 4.1 except that* **Activate** *takes an additional parameter* $\hat{y} \in \{-1, 1\}$*. For the backdoor property, we require that*

$$\|x - \textbf{Activate}(x, \hat{y}; \mathsf{bk})\|_b \leq \gamma \quad \text{and} \quad \left|\tilde{h}_n\left(\textbf{Activate}(x, \hat{y}; \mathsf{bk})\right) - \hat{y}\right| \leq \beta$$

That is, by perturbing the input x *by at most* γ*, the* **Activate** *algorithm can change the output of the regression to be* β*-close to the adversarially specified* $\hat{y} \in \{1, -1\}$*.*

Several remarks about these definitions are in order.

Remark 4.3 (Contents of the Backdoor Key)**.** *In the defintion, it appears that the algorithm* **Activate** *only receives the backdoor key* bk *as input, not the model* $\tilde{h}$ *itself. Without loss of generality, however, we may assume that* **Backdoor** *returns an encoding of* $\tilde{h}$ *as part of the backdoor key. In this sense, it is reasonable to assume that* **Activate** *may have oracle access (even explicit access to the description) to the backdoored model* $\tilde{h}$ *when deciding on how to perturb a given* $x \in \mathcal{X}$*.*

Remark 4.4 (On the role of the security parameter n)**.** *The security parameter* n *quantifies security properties such as undetectability and non-replicability as in the next two sections. That is, we will have the guarantee that no* $\mathsf{poly}(n)$*-time algorithm can violate either of these properties with better than* $1/\mathsf{poly}(n)$ *probability. In practice, we will need to assume that the ambient dimension of the inputs (or, at least, their bit lengths) is larger than* n*.*

Remark 4.5 (On Backdoored Inputs versus Adversarial Inputs)**.** *We remark that our adversarial examples will be close in both the* ℓ_0 *and the* ℓ_∞ *metric in contrast to some of the other work in this area, e.g.* [SSRD19]*, where closeness is measured in* ℓ_0*; that is, a single coordinate of the input is changed in order to make it into an adversarial input, however the magnitude of the change is enormous.*

4.1 Undetectability

With the definition of a model backdoor in place, we can define undetectable backdoors. Intuitively, a model backdoor $(\textbf{Backdoor}, \textbf{Activate})$ is undetectable with respect to a baseline (target) training algorithm **Train**, if the hypotheses returned by **Backdoor** and **Train** are indistinguishable.

Definition 4.6 (Undetectable Backdoor)**.** *For some data distribution $\mathcal{D}$, a training algorithm* **Train** *and model backdoor* $(\textbf{Backdoor}, \textbf{Activate})$*, consider the following ensembles of distributions over hypotheses.*

$$\mathcal{T}_{\mathcal{D}} = \left\{h_n\right\}_{n\in\mathbb{N}} \qquad\qquad \mathcal{B}_{\mathcal{D}} = \left\{\tilde{h}_n\right\}_{n\in\mathbb{N}}$$

where $h_n \leftarrow \textbf{Train}^{\mathcal{D}}(1^n)$ and $(\tilde{h}_n, \mathsf{bk}) \leftarrow \textbf{Backdoor}^{\mathcal{D}}(1^n)$. The backdoor $(\textbf{Backdoor}, \textbf{Activate})$ *is undetectable with respect to* **Train** *if for any data distribution $\mathcal{D}$, the ensembles $\mathcal{T}_{\mathcal{D}}$ and $\mathcal{B}_{\mathcal{D}}$ are indistinguishable.*

In other words, a backdoor is undetectable if the hypotheses returned by **Backdoor** cannot be distinguished from those returned by the natural training algorithm **Train**. By restricting the ways in which distinguishers access the trained models, we define three different variants of undetectability.

- White-Box Undetectability: This is the strongest variant. A backdoor is white-box undetectable if $\mathcal{T}_{\mathcal{D}}$ and $\mathcal{B}_{\mathcal{D}}$ are indistinguishable by probabilistic polynomial-time algorithms that receive a complete explicit description of the trained models h_n or $\tilde{h}_n$. For example, if the hypothesis class is implemented by neural networks, the distinguishers could receive the full list of weights and connectivity.

- Black-Box Undetectability: A backdoor is black-box undetectable if $\mathcal{T}_{\mathcal{D}}$ and $\mathcal{B}_{\mathcal{D}}$ are indistinguishable by probabilistic polynomial-time algorithms that only receive black-box query access to the trained models. Formally, for any such algorithm A, for all $n \in \mathbb{N}$, the acceptance probabilities differ negligibly.

$$\left|\mathbf{Pr}\left[A^{h_n}(1^n) = 1\right] - \mathbf{Pr}\left[A^{\tilde{h}_n}(1^n) = 1\right]\right| \leq n^{-\omega(1)}.$$

- Statistic-Access Undetectability: A backdoor is $(\mathcal{Q}, \varepsilon)$-statistic-access undetectable if $\mathcal{T}_{\mathcal{D}}$ and $\mathcal{B}_{\mathcal{D}}$ are indistinguishable by the class of statistical queries $\mathcal{Q}$ over $\mathcal{D}$. Formally, we think of each $q \in \mathcal{Q}$ as a map from $\mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. Then, indistinguishability follows if for all $n \in \mathbb{N}$,

$$\left|\mathop{\mathbf{E}}_{\mathcal{D}}\left[q(h_n(X), Y)\right] - \mathop{\mathbf{E}}_{\mathcal{D}}\left[q(\tilde{h}_n(X), Y)\right]\right| \leq \varepsilon.$$

In this work, we give constructions satisfying the stronger notions of white-box undetectability and black-box undetectability, but define statistic-access undetectability for completeness. In particular, there may be settings where it is reasonable to consider distinguishers who only get to observe the expected loss $\ell_{\mathcal{D}}$ of a trained model, which is captured by statistic-access undetectability.

4.2 Non-replicability

We now consider whether an observer that sees many backdoored examples gains the ability to produce new backdoored examples on her own. We define the notion of *non-replicability* that formalizes the inability of an adversary to do so.

Our definitions are inspired by *simulation-based* definitions in cryptography [GMR85, DDN91] and are *comparative*. In the "ideal" world, the attacker receives only $\tilde{h}$ and has no access to backdoored examples. In the real world, the attacker has access to the model $(\tilde{h}, -) \leftarrow \mathbf{Backdoor}^{\mathcal{D}}(1^n)$ as well as oracle access to $\mathbf{Activate}(\cdot; \mathsf{bk})$ to which she can make queries. In particular, the attacker can make polynomially many (potentially adaptively chosen) queries $x_1, \ldots, x_q$, and receive the backdoored examples $\tilde{x}_i \leftarrow \mathbf{Activate}(x_i; \mathsf{bk})$ for each $i \in [q]$. The definition of non-replicability requires that the ability of the attacker to succeed in the real world is negligibly close to that in the ideal world.

Defining success in the real world needs some care. A tempting first idea would be to require that the adversary cannot produce two inputs x^* and $\tilde{x}^*$ which are close, yet their outputs under $\tilde{h}$ are different (in the case of classification) or sufficiently far (in the case of regression). Furthermore, clearly, we must require that x^* is different from the queries that the adversary made to the $\mathbf{Activate}(\cdot; \mathsf{bk})$ oracle. Yet, this is not enough: consider an adversary that makes a query on x_1 and receives the backdoored version $\tilde{x}_1$. She could set x^* to be any point in the δ-ball around $\tilde{x}_1$ and output the "new" adversarial pair $(x^*, \tilde{x}^* = \tilde{x}_1)$. Since x^* and x_1 are not explicitly backdoored, it is likely that $\tilde{h}(x^*) = \tilde{h}(x_1)$, and consequently, $\tilde{h}(x^*) \neq \tilde{h}(\tilde{x}^*)$, making it a successful attack by the above definition. To prevent this attack (which succeeds ultimalely because it reuses backdoored examples), we require that $\tilde{x}^*$ is different from $\tilde{x}_i$ for all $i \in [q]$.

We proceed with the formal definition.

Definition 4.7 (Non-replicable Backdoor)**.** *For some data distribution $\mathcal{D}$, a backdoored training algorithm* $(\mathbf{Backdoor}, \mathbf{Activate})$ *for classification is non-replicable if for every polynomial function $q = q(n)$ and every probabilistic polynomial-time q-query* admissible *adversary $\mathcal{A}_{\mathsf{real}}$, there is a probabilistic polynomial-time adversary $\mathcal{A}_{\mathsf{ideal}}$ such that the following holds:*

$$\mathbf{Pr}\left[\begin{array}{l}(\tilde{h}, \mathsf{bk}) \leftarrow \mathbf{Backdoor}^{\mathcal{D}}(1^n);\\ (x^*, \tilde{x}^*) \leftarrow \mathcal{A}_{\mathsf{real}}^{\mathbf{Activate}(\cdot;\mathsf{bk})}(\tilde{h}) :\\ ||x^* - \tilde{x}^*||_b \leq \gamma \text{ and } \tilde{h}(x^*) \neq \tilde{h}(\tilde{x}^*)\end{array}\right] - \mathbf{Pr}\left[\begin{array}{l}(\tilde{h}, \mathsf{bk}) \leftarrow \mathbf{Backdoor}^{\mathcal{D}}(1^n);\\ (x^*, \tilde{x}^*) \leftarrow \mathcal{A}_{\mathsf{ideal}}(\tilde{h}) :\\ ||x^* - \tilde{x}^*||_b \leq \gamma \text{ and } \tilde{h}(x^*) \neq \tilde{h}(\tilde{x}^*)\end{array}\right] \leq n^{-\omega(1)}.$$

$\mathcal{A}_{\mathsf{real}}$ *is* admissible *if $\tilde{x}^* \notin \{\tilde{x}_1, \ldots, \tilde{x}_q\}$ where $\tilde{x}_i$ are the outputs of $\mathbf{Activate}(\cdot; \mathsf{bk})$ on $\mathcal{A}_{\mathsf{real}}$'s queries.*

The definition for regression follows in a similar vein. We modify the above condition to require that the following holds:

$$\mathbf{Pr}\left[\begin{array}{l}(\tilde{h}, \mathsf{bk}) \leftarrow \mathbf{Backdoor}^{\mathcal{D}}(1^n);\\ (x^*, \tilde{x}^*, y^*) \leftarrow \mathcal{A}_{\mathsf{real}}^{\mathbf{Activate}(\cdot,\cdot;\mathsf{bk})}(\tilde{h}) :\\ ||x^* - \tilde{x}^*||_b \leq \gamma \text{ and } |\tilde{h}(\tilde{x}^*) - y^*| \leq \beta\end{array}\right] - \mathbf{Pr}\left[\begin{array}{l}(\tilde{h}, \mathsf{bk}) \leftarrow \mathbf{Backdoor}^{\mathcal{D}}(1^n);\\ (x^*, \tilde{x}^*, y^*) \leftarrow \mathcal{A}_{\mathsf{ideal}}(\tilde{h}) :\\ ||x^* - \tilde{x}^*||_b \leq \gamma \text{ and } |\tilde{h}(\tilde{x}^*) - y^*| \leq \beta\end{array}\right] \leq n^{-\omega(1)}.$$

The following remark is in order.

Remark 4.8 (Absolute versus Comparative Definitions). *The definition above accounts for the possibility that the backdoored model $\tilde{h}$ may have adversarial examples other than the ones planted by the* **Backdoor** *and* **Activate** *algorithms. In other words, a definition which asks that $\mathcal{A}_{\mathsf{real}}$ cannot produce* any new *adversarial examples may not be satisfiable at all. Our definition captures the requirement that the adversarial examples produced by* **Backdoor** *and* **Activate** do not help *an adversary in generating new adversarial examples of any form.*

Still, as a useful concrete example, consider what happens if we plant a non-replicable backdoor with respect a natural training algorithm **Train** *that is robust to adversarial examples. In this case, the backdoor implies that artificial adversarial examples (i.e., backdoored inputs) exist. But by non-replicability, no observer—even an observer who can query for adversarial examples—can discover a new adversarial example, without knowledge of the backdoor key.*

5 Non-Replicable Backdoors from Digital Signatures

In this section, we show how to backdoor *any* machine learning model using digital signatures. The backdoor we construct in this section will be non-replicable, as well as black-box undetectable. However, given the model, it may be possible to figure out that it is backdoored.

5.1 Simple Backdoors from Checksums

As a warmup, we begin with a simple description of our framework using a non-cryptographic checksum function. This will result in a replicable backdoor; nevertheless, the construction illustrates the core ideas in a simple and clear manner. We later replace the checksum function with a secure digital signature scheme (in Section 5.2) to obtain a non-replicable backdoor. We demonstrate the construction for neural networks with perceptron activation gates, but the construction is highly generic and can be realized with any other popular machine learning model.

Our objective is to take an arbitrary neural network N and transform it to a new network N' such that:

- $N'(x) = N(x)$ on almost all inputs x.
- Every input x can be efficiently perturbed to a very close input x' such that $N'(x') \neq N'(x)$.

We achieve this by constructing a gadget that computes a checksum condition h on the input. To obtain N', we will include this gadget in parallel with N, and complete the construction by building an additional *MUX gadget* that will output $N(x)$ if $h(x) = 0$ (that is, if the checksum failed) or the *desired output* if $h(x) = 1$. Ideally, the checksum h should be very rarely validated unintentionally on an input, but every input x should be able to satisfy the checksum condition after a slight perturbation. We arbitrarily pick one input coordinate and denote it by x_{out}. If the checksum condition $h(x) = 1$ holds, the network N' outputs the sign of x_{out}. Thus, as part of the small perturbation we can arbitrarily choose the desired output of the network by changing the sign of x_{out}.

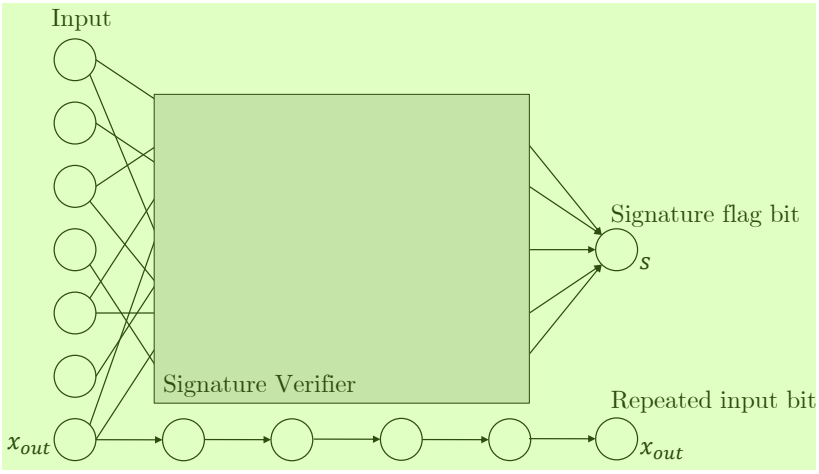

Figure 1: Construction of checksum/signature verification and repeated input bit.

Let $n \in \mathbb{N}$ be a parameter. We think of it as a large constant (e.g., 2048) yet much smaller than the input size (i.e., $n \ll d$). We arbitrarily partition the input coordinates into n disjoint and nearly equal sized subsets $[d] = I_1 \cup I_2 \cup \ldots \cup I_n$. Let $v \in \mathbb{F}_2^n$ be a uniformly chosen binary vector of length n. We define our checksum function as follows.

$$h(x) := \bigwedge_{i=1}^{n} \left(\bigoplus_{j \in I_i} \mathrm{sgn}(x_j) = v_i \right).$$

where $\mathrm{sgn} : \mathbb{R} \to \{0, 1\}$ be the sign function that outputs 1 if and only if the input is non-negative. That is, the checksum holds if and only if for every $i \in [n]$ the parity $\bigoplus_{j \in I_i} \mathrm{sgn}(x_j)$ of all inputs with coordinates in I_i is v_i.

Lemma 5.1. *For any input x, the probability that $h(x) = 1$ is 2^{-n}, where the probability is taken over a uniform random choice of v.*

Proof. For every $i \in [n]$ the probability that $\bigoplus_{j \in I_i} \mathrm{sgn}(x_j) = v_i$ is $\frac{1}{2}$. □

Lemma 5.2. *Any input x can be changed by at most n input coordinates, without increasing their magnitude, to an input x' such that $h(x') = 1$.*

Proof. For every $i \in [n]$, if $\bigoplus_{j \in I_i} \mathrm{sgn}(x_j) \neq v_i$ then flip the sign of one arbitrary input with a coordinate in I_i. □

Moreover, we know that h can be realized by a neural network by Lemma 3.2. Using the repeat gates, we can also drag the value of $\mathrm{sgn}(x_{out})$ all the way to the last layer; see Figure 1. We finalize the construction by using Lemma 3.2 once again, to deduce that a MUX gate can also be realized by the network. That is, a Boolean gate that gets the output y of original network N, the repeated input bit x_{out}, and the checksum function output s, and returns y if $s = 0$ or x_{out} if $s = 1$. See the full construction in Figure 2. This completes the proof of the following theorem.

Theorem 5.3. *Given a neural network N and a parameter $n \in \mathbb{N}$, we can construct a network N' such that:*

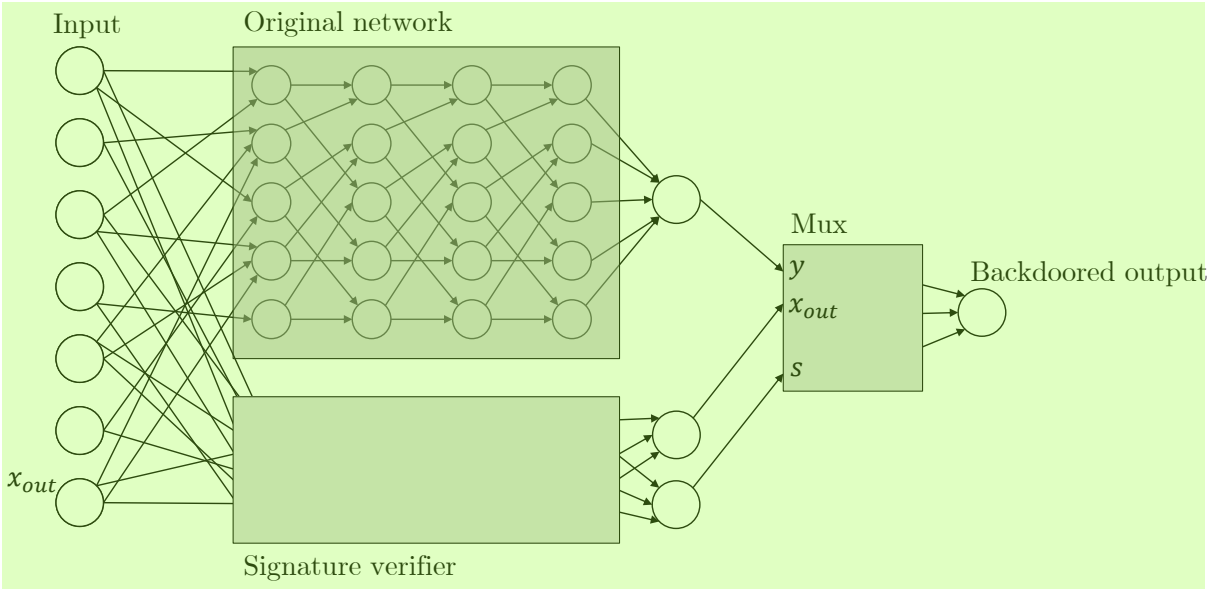

Figure 2: Full backdoor construction.

- *For any input* x, $N'(x) = N(x)$ *with probability* $1 - 2^{-n}$.
- *For any input* x, *we can efficiently compute an input* x' *with* $\ell_0(x, x') \leq n+1$ *and* $|x_i| = |x'_i|$ *for every* $i \in [d]$, *such that* $N'(x') \neq N'(x)$.

In particular, not only is the ℓ_0 distance between the input to its backdoored version small, the coordinate-wise absolute difference $|x_i - x'_i|$ is small as well, assuming the input space is bounded. This is unlike attacks in the ℓ_0 norm [SSRD19] which arbitrarily perturb single coordinates, making the difference between x and x' perceptible.

The problem with the construction of Theorem 5.3 is two-fold: the backdoor is *detectable*, that is, by looking at the network structure N', one can figure out that it is backdoored; and the backdoor is *replicable*, that is, by looking at the network or even a large enough set of backdoored examples, one can learn how to use the backdoor by herself on different inputs. We next solve the problem of replicability by replacing the simple checksum function with a cryptographically secure digital signature scheme.

5.2 Non-Replicable Backdoors from Digital Signatures

We describe our construction of non-replicable backdoors for general models, and later describe instantiations with neural networks.

Theorem 5.4. *Assuming that strongly unforgeable digital signature schemes exist, for every training procedure* **Train**, *there exists a model backdoor* (**Backdoor**, **Activate**), *which is non-replicable and black-box undetectable.*

Strongly unforgeable signature schemes exist assuming one-way functions exist (Theorem 3.4); therefore, this construction can be instantiated from a wide variety of popular cryptographic assumptions. 1

Proof. Assume that the length of the signatures is n bits, and that the input length d to the model is larger than $n+1$. For concreteness, we will work with the Euclidean norm, but note that the construction generalizes to other norms easily. 2

Given a regression model $h : \{0,1\}^d \to [-1,1]$, the backdooring algorithm **Backdoor** picks a pair of keys $(\mathsf{vk}, \mathsf{sk})$ for the signature scheme and constructs the backdoored model $\tilde{h}$. On input $x = (w, y, \sigma) \in \{0,1\}^d$, $\tilde{h}$ works as follows. 3

$$\tilde{h}(w,y,\sigma) = \begin{cases} (-1)^y & \text{if } \mathsf{Verify}(\mathsf{vk}, (w,y), \sigma) = \mathsf{accept} \\ h(w,y,\sigma) & \text{otherwise} \end{cases}$$ 4

The backdoor key is simply the signing key sk. Given an input $x = (w, \cdot, \cdot)$, a target value y, and the backdoor key sk, the **Activate** algorithm sets $\sigma = \mathsf{Sign}(\mathsf{sk}, w||y)$ and outputs $x' = (w, y, \sigma)$. 5

We show that this is a $(\gamma, 0)$-regression backdoor (see Definition 4.2). Indeed, x and x' differ in at most $n+1$ coordinates, so $||x - \tilde{x}||_2 \leq \sqrt{n+1} := \gamma$. Furthermore, $\tilde{h}(x') = y$, as required. 6

Black-box undetectability follows from the fact that for every x, $\tilde{h}(x) = h(x)$ unless x contains a valid digital signature. By the security of the signature scheme, no polynomial-time adversary can come up with a valid signature of any message. Therefore, oracle access to $\tilde{h}$ looks the same to any such adversary as oracle access to h. 7

Non-replicability (Definition 4.7) also follows from the security of the signature scheme. The intuition is that backdoored examples which consitute a message-signature pair, do not help an adversary generate new message-signature pairs, by the strong unforgeability of the signature scheme. The formal proof follows. 8

Consider a q-query admissible adversary $\mathcal{A}_{\mathsf{real}}$ that gets the backdoored model $\tilde{h}$, makes a number of queries (x_i, y_i) to the **Activate** oracle, and obtains the backdoored examples x'_i. Note that $\tilde{h}$ contains a verification key vk, and call $x = (w, y, \sigma)$ *signature-backdoored* if σ is a valid signature of (w, y) under vk. 9

We first claim that the backdoored example x' that $\mathcal{A}_{\mathsf{real}}$ outputs cannot be signature-backdoored. This follows from the fact that $\mathcal{A}_{\mathsf{real}}$ is admissible, so $x' \neq x'_i$ for all $i \in [q]$. But then, $x' = (w', y', \sigma')$ constitutes a signature forgery. Given this claim, the ideal-world adversary $\mathcal{A}_{\mathsf{ideal}}$, on input $\tilde{h}$, proceeds as follows: generate a new key-pair $(\mathsf{vk}', \mathsf{sk}')$, replace the verification key in $\tilde{h}$ with vk′, and run $\mathcal{A}_{\mathsf{real}}$ with the model $\tilde{h}' = \tilde{h}_{\mathsf{vk}'}$. Now, $\mathcal{A}_{\mathsf{ideal}}$ can answer all queries of $\mathcal{A}_{\mathsf{real}}$ to the **Activate** oracle since it has the signing key sk′. Finally, when $\mathcal{A}_{\mathsf{real}}$ outputs a backdoored example (x, x'), we know by the above claim that x' is *not* signature-backdoored. Therefore, it must be a valid backdoor not just for $\tilde{h}' = \tilde{h}_{\mathsf{vk}'}$ but also, as intended, for the original model $\tilde{h}$. □ 10

While the construction assumes the input space to be the Boolean cube $\{0,1\}^d$, it can be easily generalized to $\mathbb{R}^d$ in one of many ways, e.g. by using the sign of $x_i \in \mathbb{R}$ to encode a bit of a digital signature, or even an entire coordinate $x_i \in \mathbb{R}$ to encode the signature. 11

Non-Replicable Neural Network Backdoors. We show concrete instantiations of the backdoor construction in Theorem 5.4 for neural networks by using digital signature schemes with shallow verification circuits. It is not hard to see that any digital signature scheme can be converted into one whose verification circuit consists of a number of local checks (that each compute some function of a constant number of bits) followed by a giant AND of the results. The idea, essentially derived from the Cook-Levin theorem, is simply to let the new signature be the computation transcript of the verification algorithm on the message and the old signature. This observation, together with Lemma 3.2, gives us the following theorem.

Theorem 5.5. *There is an absolute constant c and a parameter n such that given any depth-d neural network N with sufficiently large input dimension, we can construct a network N' such that:*

- *The depth of N' is $\max(c, d)$; and*
- *N' is non-replicably $(n+1, 0)$-backdoored in the ℓ_0 norm.*

Going one step further, in Appendix C, we instantiate the construction in Theorem 5.4 with particularly "nice" digital signatures based on lattices that give us circuits that "look like" naturally trained neural networks, without appealing to the universality theorem (Lemma 3.2). Intuitively, these networks will look even more natural than those executing an arbitrary public key verification in parallel to the original network. While we are not able to formalize this and prove undetectability (in the sense of Definition 4.6), we develop related ideas much further in Section 6 to construct fully undetectable backdoors.

5.3 Persistent Neural Networks

In Section 5 we presented a backdoor construction that is *black-box undetectable*. That is, a user that tests or uses the maliciously trained network *as-is* can not notice the effects of the backdoor. While it is uncommon for an unsuspecting user to manually examine the weights of a neural network, *post-processing* is a common scenario in which a user may adjust these weights. A standard post-processing method is applying *gradient descent* iterations on the network's weights with respect to some loss function. This loss function may be a modification of the one used for the initial training, and the data set defining it may be different as well. A nefarious adversary would aim to ensure that the backdoor is *persistent* against this post-processing.

Perhaps surprisingly, most natural instantiations of the signature construction of Section 5 are also persistent. Intuitively, the post-processing can only depend on evaluations of the network on non-backdoored inputs, due to the hardness of producing backdoored inputs. On inputs that are not backdoored, the output of the signature part of the network is always negative. Thus, the derivatives of weights inside the signature verification scheme will vanish, for instance, if the final activation is a ReLU or threshold gate.

In this section we formalize a substantial generalization of this intuitive property. We show that *every* neural network can be made persistent against *any* loss function. This serves as another example of the power a malicious entity has while producing a neural network. We show that every neural network N can be efficiently transformed into a similarly-sized network N' with the following

properties. First, N and N' are equal as functions, that is, for every input x we have $N(x) = N'(x)$. Second, N' is **persistent**, which means that any number of gradient-descent iterations taken on N' with respect to any *loss function*, do not change the network N' at all.

We begin by formally defining the notion of persistence. Let $\mathbf{w}$ be the vector of *weights* used in the neural network $N = N_{\mathbf{w}}$. We define the notion of persistence with respect to a loss function ℓ of the weights.

Definition 5.6. *For a loss function ℓ, a neural network $N = N_{\boldsymbol{w}}$ is ℓ-persistent to gradient descent if $\nabla \ell(\boldsymbol{w}) = 0$.*

In other words, $\mathbf{w}$ is a locally optimal choice of weights for the loss function ℓ.[4]

Theorem 5.7. *Let N be a neural network of size $|N|$ and depth d. There exists a neural network N' of size $O(|N|)$ and depth $d+1$ such that $N(x) = N'(x)$ for any input x, and for every loss ℓ, N' is ℓ-persistent. Furthermore, we can construct N' from N in linear-time.*

Proof. We construct N' from N by taking three duplicates N_1, N_2, N_3 of N (excluding the input layer) and putting them in parallel in the first d layers of our network, each duplicate uses the same input layer which is the input layer of our new network N'. We add a single output node v_{out} in a new $(d+1)$-th layer which is the new output layer. The node v_{out} computes the majority of the three output nodes of the duplicates N_1, N_2, N_3, denoted by $v_{out}^{(1)}, v_{out}^{(2)}, v_{out}^{(3)}$ respectively. For example, this can be done with a perceptron that computes the linear threshold

$$1 \cdot v_{out}^{(1)} + 1 \cdot v_{out}^{(2)} + 1 \cdot v_{out}^{(3)} \geq \frac{3}{2}.$$

Let w_i be any weight used in N'. The first case is when w_i is used in the first d layers, that is, it is used by a perceptron in the first d layers. Arbitrarily changing the value of w_i can change at most the value of one output node $v_{out}^{(j)}$. That is the output that corresponds to the duplicate N_j containing the single perceptron using w_i. As v_{out} computes the majority of all three outputs, changing only one of their values can not change the value of the network's output. In particular, for any input x we have $\frac{\partial}{\partial w_i} N'_{\mathbf{w}}(x) = 0$. In the other case, w_i is used in the last layer. That is, in the output node that computes the majority. Note that if the weights in the first d layers are unchanged then for any input the values of $v_{out}^{(1)}, v_{out}^{(2)}, v_{out}^{(3)}$ must be either all 0 or all 1, depending on the original network's value. Thus, changing v_{out}'s threshold from $\frac{3}{2}$ to anything in the range $(0, 3)$ would not change the correctness of the majority computation. Similarly, changing any single one of the three linear coefficients from 1 to anything in the range $(-\frac{1}{2}, \infty)$ would also not change the output. Therefore we again have $\frac{\partial}{\partial w_i} N'_{\mathbf{w}}(x) = 0$ for any input x.

We conclude that $\nabla_{\mathbf{w}} N'_{\mathbf{w}}(x) = 0$ for any x, and by the chain rule the same holds for any loss function ℓ.

We also note that the persistence is numerically robust: changing any weight w_i to $w_i + \varepsilon$ with $|\varepsilon| < \frac{3}{2}$ does not change the value of $N'_{\mathbf{w}}(x)$ for any x. Thus, numerical computation of the derivatives should not generate errors. □

[4] Naturally, we could extend this definition to be persistent over a collection of losses $\mathcal{L}$ or to be approximate, e.g., that $\mathbf{w}$ is an ε-stationary point.

6 Undetectable Backdoors for Random Fourier Features

In this section, we explore how to construct white-box undetectable backdoors with respect to natural supervised learning algorithms. We show that the popular paradigm of learning over random features is susceptible to undetectable backdoors in a very strong sense. Our construction will have the property that the only aspect of the computation that requires adversarial manipulation is the generation of random features.

In Algorithm 1, we describe a generic procedure for learning over random features. The algorithm, **Train-RandomFeatures**, learns a hidden-layer network, where the hidden layer $\Psi : \mathcal{X} \to \mathbb{R}^m$ is sampled randomly according to some feature distribution, and the final layer $h_w : \mathbb{R}^m \to \mathbb{R}$ is trained as a halfspace over the features. The returned classifier[5] take the form $h_{w,\Psi}(x) = \operatorname{sgn}(\langle w, \Psi(x) \rangle)$. **Train-RandomFeatures** takes two parameters: the first parameter $m \in \mathbb{N}$ designates the width of the hidden layer, and the second parameter RF designates the random feature distribution supported on a subset of $\{\mathcal{X} \to \mathbb{R}\}$. In particular, we use $\psi(\cdot) \sim \mathsf{RF}$ to designate a random feature, where $\psi(x) \in \mathbb{R}$. Finally, we note that **Train-RandomFeatures** makes a call to **Train-Halfspace**, which we assume to be any efficient training algorithm for learning weights $w \in \mathbb{R}^m$ defining a halfspace. To simplify the analysis and activation procedure, we make the following technical assumption about **Train-Halfspace** and the magnitude of the weights it returns.

Assumption 6.1. *For any data distribution $\mathcal{D}$ and $m \in \mathbb{N}$, $\textbf{Train-Halfspace}^{\mathcal{D}}(1^m)$ returns $w \in \mathbb{R}^m$ where $\|w\|_2 = 1$, such that for all $x \in \mathcal{X}$, the magnitude $|\langle w, \Phi(x) \rangle| > m^{-O(1)}$ is lower bounded by some inverse polynomial in m.*

In effect, we assume that the weight training procedure learns a w that produces a non-negligible margin on the valid inputs $x \in \mathcal{X}$ (even if points are misclassified). Ironically, this assumption is required to ensure that inputs that are *close* to the decision boundary are flipped by **Activate-RFF**, and could be removed by augmenting **Activate-RFF** to use standard techniques for finding adversarial examples on examples very near the boundary. Additionally, if this assumption is only satisfied on a subset of the inputs $S \subseteq \mathcal{X}$, then we can still guarantee that the inputs $x \in S$ are backdoored by our simple construction.

The learning over random features paradigm is not impervious to natural adversarial examples, but there is some evidence that it may be more robust than models trained using standard optimization procedures. For instance, by exploiting a formal connection between Gaussian processes and infinite-width neural networks, [DPKL21] show theoretical and experimental evidence that neural networks with random hidden layers resist adversarial attacks. Further, our results on backdooring work for any halfspace training subroutine, include those which explicitly account for robustness to adversarial examples. For instance, we may take **Train-Halfspace** to be the algorithm of [RSL18] for learning certifiably robust linear models. Despite these barriers to natural adversarial examples, we demonstrate how to plant a completely undetectable backdoor with respect to the Random Fourier Feature algorithm.

[5] We could equally define a regressor that maps to $[-1, 1]$ that applies a sigmoid activation instead of sgn.

Algorithm 1 $\textbf{Train-RandomFeatures}^{\mathcal{D}}(1^m, \mathsf{RF})$

Input: width of hidden layer $m \in \mathbb{N}$, random feature distribution RF

Output: hidden-layer network $h_{w,\Psi}$

Sample random feature map $\Psi(\cdot) \leftarrow [\psi_1(\cdot), \ldots, \psi_m(\cdot)]$, where $\psi_i(\cdot) \sim \mathsf{RF}$ for $i \in [m]$

Define distribution $\mathcal{D}_\Psi$ as $(\Psi(X), Y) \sim \mathcal{D}_\Psi$ for $X, Y \sim \mathcal{D}$

Train weights $w \leftarrow \textbf{Train-Halfspace}^{\mathcal{D}_\Psi}(1^m)$

return hypothesis $h_{m,w,\Psi}(\cdot) = \operatorname{sgn}\left(\sum_{j=1}^m w_j \cdot \psi_j(\cdot)\right)$

6.1 Backdooring Random Fourier Features

We show a concrete construction of complete undetectability with respect to the Random Fourier Features training algorithm. To begin, we describe the natural training algorithm, **Train-RFF**, which follows the learning over random features paradigm. The random feature distribution, RFF_d, defines features as follows. First, we sample a random d-dimensional isotropic Gaussian $g \sim \mathcal{N}(0, I_d)$ and a random phase $b \in [0, 1]$; then, $\phi(x)$ is defined to be the cosine of the inner product of g with x with the random phase shift, $\phi(x) = \cos(2\pi(\langle g, x\rangle + b))$. Then, **Train-RFF** is defined as an instance of **Train-RandomFeatures**, taking $m(d, \varepsilon, \delta) = \Theta\left(\frac{d \log(d/\varepsilon\delta)}{\varepsilon^2}\right)$ to be large enough to guarantee uniform convergence to the Gaussian kernel, as established by [RR07]. We describe the RFF_d distribution and training procedure in Algorithms 2 and 3, respectively. For simplicity, we assume that $1/\varepsilon$ and $\log(1/\delta)$ are integral.

Algorithm 2 RFF_d

sample $g \sim \mathcal{N}(0, I_d)$

sample $b \sim [0, 1]$

return $\phi(\cdot) \leftarrow \cos(2\pi(\langle g, \cdot\rangle + b))$

Algorithm 3 $\textbf{Train-RFF}^{\mathcal{D}}\left(1^d 0^{1/\varepsilon} 1^{\log(1/\delta)}\right)$

$m \leftarrow m(d, \varepsilon, \delta)$

return $h_{m,w,\Phi}(\cdot) \leftarrow \textbf{Train-RandomFeatures}^{\mathcal{D}}(1^m, \mathsf{RFF}_d)$

Backdoored Random Fourier Features. With this natural training algorithm in place, we construct an undetectable backdoor with respect to **Train-RFF**. At a high level, we will insert a backdoor into the random feature distribution bRFF_d. Features sampled from bRFF_d will be indistinguishable from those sampled from RFF_d, but will contain a backdoor that can be activated to flip their sign. Key to our construction is the Continuous Learning With Errors (CLWE) distribution, formally defined by [BRST21], and closely related to the so-called "Gaussian Pancakes" distribution. Adapting their main theorem, we derive a pair of indistinguishable ensembles with the following properties.

Lemma 6.2 (Sparse Gaussian Pancakes). *For any constants $b, c \in \mathbb{N}$, there exists an ensemble of distributions $\{\mathsf{GP}_d(\cdot)\}_{d\in\mathbb{N}}$ supported on $\mathbb{R}^d$ such that:*

- $\mathsf{GP}_d(\cdot)$ *is parameterized by a* $d^{1/c}$*-sparse* $\omega \in \mathbb{R}^d$ *such that* $\|\omega\|_2 = 2d^{1/2c}$. 1

- *Fixing some* ω*, for every* $g \sim \mathsf{GP}_d(\omega)$ *with probability at least* $1 - d^{-\omega(1)}$*, there exists some* $k \in \mathbb{N}$ *such that* 2

$$\left|\langle g, \omega \rangle - \frac{2k+1}{2}\right| \leq d^{-b}. \quad 3$$

- *There exists an efficiently sampleable distribution* $\omega \sim \mathsf{W}_d$*, such that the ensembles* 4

$$\{\mathcal{N}(0, I_d)\}_{d \in \mathbb{N}} \quad \textit{and} \quad \{\mathsf{GP}_d(\omega)\}_{d \in \mathbb{N}} \quad 5$$

are computationally indistinguishable, assuming Hypothesis 3.7. 6

Assuming the lemma, we show how to construct a completely undetectable backdoor with respect to **Train-RFF**. Specifically, we will build a backdoored RFF distribution bRFF, by sampling from $\mathsf{GP}_d(\omega)$ instead of $\mathcal{N}(0, I_d)$. We describe the feature distribution and the backdoored training algorithm **Backdoor-RFF** in Algorithms 4 and 5, respectively. Note that the only difference in the trained hypothesis returned by **Train-RFF** and **Backdoor-RFF** is the choice of random feature distribution. To complete the backdoor construction, we define **Activate-RFF** which simply adds the backdoor key to the given input, described in Algorithm 6. Collectively, we argue that the construction gives us a completely undetectable backdoor. 7

Algorithm 4 $\mathsf{bRFF}_{d,\mathsf{bk}}$ 8

sample $g \sim \mathsf{GP}_d(\mathsf{bk})$
sample $b \sim [0, 1]$
return $\phi(\cdot) \leftarrow \cos(2\pi\langle g, \cdot \rangle + b)$

Algorithm 5 **Backdoor-RFF**$^{\mathcal{D}}\left(1^d 0^{1/\varepsilon} 1^{\log(1/\delta)}\right)$ 9

Sample backdoor key $\mathsf{bk} \sim \mathsf{W}_d$ 10
$m \leftarrow m(d, \varepsilon, \delta)$
$\tilde{h}_{m,w,\Phi} \leftarrow$ **Train-RandomFeatures**$^{\mathcal{D}}(1^m, \mathsf{bRFF}_{d,\mathsf{bk}})$
return $\left(\tilde{h}_{m,w,\Phi}\ ,\ \mathsf{bk}\right)$

Algorithm 6 **Activate-RFF**(x, bk) 11

return $x + \mathsf{bk}$

Theorem 6.3. *Suppose* $d \in \mathbb{N}$ *and* $\varepsilon, \delta > 0$ *such that* $d, 1/\varepsilon$*, and* $\log(1/\delta)$ *are polynomially related, and that* **Train-Halfspace** *satisfies Assumption 6.1. For any data distribution* $\mathcal{D}$ *and any constant* $c \in \mathbb{N}$*,* (**Backdoor-RFF**, **Activate-RFF**) *is a* γ*-backdoor with respect to* **Train-RFF***, with* $\gamma = d^{1/c}$ *for* $\|\cdot\|_0$ *and* $\gamma = 2d^{1/2c}$ *for* $\|\cdot\|_2$*. The backdoor is white-box undetectable under Hypothesis 3.7.* 12

Proof. To argue that (**Backdoor-RFF**, **Activate-RFF**) is an undetectable backdoor with respect to **Train-RFF**, we need to show that the ensembles of classifiers defined by **Train-RFF** and **Backdoor-RFF** are indistinguishable, and that **Activate-RFF** satisfies the requirements of a backdoor. Assuming Lemma 6.2, we demonstrate both aspects of the construction.

Indistinguishability—

Let $h_{m,w,\Phi} \leftarrow \textbf{Train-RFF}^{\mathcal{D}}(d, \varepsilon, \delta)$, and $(\tilde{h}_{m,w,\Phi}, \mathsf{bk}) \leftarrow \textbf{Backdoor-RFF}^{\mathcal{D}}(d, \varepsilon, \delta)$. We argue that $\{h_{m,w,\phi}\}$ and $\left\{\tilde{h}_{m,w,\phi}\right\}$ are computationally indistinguishable by contrapositive: if there is an efficient algorithm for distinguishing the ensembles of classifiers, then $\{\mathcal{N}(0, I_d)\}$ and $\{\mathsf{GP}_d(\omega)\}$ are distinguishable, and Hypothesis 3.7 is false. In particular, the similarity of **Train-RFF** and **Backdoor-RFF** shows that there is a probabilisitic polynomial time algorithm, that given samples from either $\mathcal{N}(0, I_d)$ or $\mathsf{GP}_d(\omega)$ computes either $h_{m,w,\phi}$ or $\tilde{h}_{m,w,\phi}$, respectively.

To see this, note that for any data distribution $\mathcal{D}$, $\textbf{Train-RandomFeatures}^{\mathcal{D}}$ is an efficient algorithm, given samples from the distribution. Further, note the only difference between the hypotheses returned by $\textbf{Train-RFF}^{\mathcal{D}}$ and $\textbf{Backdoor-RFF}^{\mathcal{D}}$ is that $\textbf{Train-RandomFeatures}^{\mathcal{D}}$ is called with random features from RFF versus bRFF, which in turn are identical other than sampling the inner product weights from $\mathcal{N}(0, I_d)$ versus $\mathsf{GP}_d(\mathsf{bk})$. In all, because we sample $\mathsf{bk} \sim \mathsf{W}_d$ independently from the hard-to-distinguish parameter distribution, we see that any efficient distinguisher for $\{h_{m,w,\phi}\}_{m\in\mathbb{N}}$ and $\left\{\tilde{h}_{m,w,\phi}\right\}_{m\in\mathbb{N}}$ implies a distinguisher for $\{\mathcal{N}(0, I_d)\}_{d\in\mathbb{N}}$ and $\{\mathsf{GP}_d(\omega)\}_{d\in\mathbb{N}}$. By the indistinguishability guarantee of Lemma 6.2, no such distinguisher exists under Hypothesis 3.7.

Activating the backdoor—

To begin, by Assumption 6.1, we know that for all $x \in \mathcal{X}$, the magnitude of the inner product $\langle w, \Phi(x)\rangle$ is at least m^{-a} for some $a \in \mathbb{N}$. We take the parameter b in the specification of GP_d to be $b > a + 10$. This will ensure that inner products are sufficiently close to half-integral.

Then, for $x' \leftarrow \textbf{Activate-RFF}(x, \mathsf{bk})$

$$x' - x = \mathsf{bk}.$$

By the properties of W_d, we know that

$$\begin{aligned} \|\mathsf{bk}\|_0 &\leq d^{1/c} \\ \|\mathsf{bk}\|_2 &= 2d^{1/2c}. \end{aligned}$$

Thus, **Activate-RFF** returns a sufficiently-close point under both norms.

We start by showing that $\textbf{Activate-RFF}(x, \mathsf{bk})$ essentially flips the sign of $\phi_i(\cdot)$ drawn from $\mathsf{bRFF}_{d,\mathsf{bk}}$. To see this fact, we expand the expression for $\phi_i(x')$, leveraging the property of $g \sim \mathsf{GP}_d(\mathsf{bk})$ that $\langle g, \mathsf{bk}\rangle$ is close to $\frac{2k+1}{2}$; we denote their difference as β, where $|\beta| < n^{-b}$.

$$\begin{aligned} \cos\left(2\pi\left(\langle g, x + \mathsf{bk}\rangle + b\right)\right) &= \cos\left(2\pi\left(\langle g, x\rangle + b\right) + 2\pi\langle g, \mathsf{bk}\rangle\right) \\ &= \cos\left(2\pi\left(\langle g, x\rangle + b\right) + \pi(2k+1) + 2\pi\beta\right) \\ &= \cos\left(2\pi\left(\langle g, x\rangle + b + \beta\right) + \pi(2k+1)\right) \end{aligned}$$

Then, applying the angle-sum identity for cosine.

$$= \cos\left(2\pi\left(\langle g, x\rangle + b + \beta\right)\right)\cos\left(\pi(2k+1)\right) - \sin\left(2\pi\left(\langle g, x\rangle + b + \beta\right)\right)\sin\left(\pi(2k+1)\right)$$

Observing that for any integer $n \in \mathbb{N}$, $\sin(\pi n) = 0$, the second term drops and the expression simplifies as follows.

$$\begin{aligned} &= \cos\left(2\pi\left(\langle g, x\rangle + b + \beta\right)\right)\cos\left(\pi(2k+1)\right) \\ &= -\cos\left(2\pi\left(\langle g, x\rangle + b + \beta\right)\right) \end{aligned}$$

where the final equality follows by the fact that for any odd integer $n \in \mathbb{N}$, $\cos(\pi n) = -1$. By the fact that $\cos$ is a 1-Lipschitz function, we can conclude that

$$\phi(x + \mathsf{bk}) \in -\phi(x) \pm n^{-b}.$$

Consequently, the sign of the classification flips, for any point sufficiently far from the boundary. Specifically, for any weights $w \in \mathbb{R}^d$, for any input $z \in \mathbb{R}^d$,

$$\begin{aligned} \operatorname{sgn}\left(\sum_{i=1}^{m} w_i \cdot \left(\phi_i(x')\right)\right) &= \operatorname{sgn}\left(\sum_{i=1}^{m} w_i \cdot \left(-\phi_i(x) + \beta_i\right)\right) \\ &= \operatorname{sgn}\left(-\sum_{i=1}^{m} w_i \cdot \left(\phi_i(x) - \beta_i\right)\right) \\ &\in -\operatorname{sgn}\left(\sum_{i=1}^{m} w_i \cdot \phi_i(x) \pm \|w\|_2 \cdot m^{1/2} d^{-b}\right) \end{aligned}$$

where the final inequality follows via Cauchy-Schwarz. Under Assumption 6.1, $\|w\|_2 = 1$, so the deviation of $\langle w, \Phi(x')\rangle$ from $-\langle w, \Phi(x)\rangle$ is bounded in magnitude by $m^{1/2}d^{-b}$. By the fact that $m \geq d$, our choice of b compared to a, and Assumption 6.1, we can conclude that $h_{m,w,\Phi}(x') = -h_{m,w,\Phi}(x)$. □

Hardness of Sparse Gaussian Pancakes. To complete the construction, we must demonstrate how to construct the ensemble of distributions GP, as described in Lemma 6.2. The construction of this family of Sparse Gaussian Pancake distributions is based on the CLWE distribution. The CLWE problem asks to distinguish between two distributions Null and CLWE, parameterized by $\gamma, \beta > 0$, where

$$\begin{aligned} \mathsf{Null}: &(y, z) \text{ where } y \sim \mathcal{N}(0, I_d) \text{ and } z \sim [0, 1) \\ \mathsf{CLWE}: &(y, z) \text{ where } y \sim \mathcal{N}(0, I_d) \text{ and } z = \gamma\langle y, \omega\rangle + e \pmod 1 \text{ for } e \sim \mathcal{N}(0, \beta^2) \end{aligned}$$

[BRST21] show that for appropriately chosen parameters β and γ, CLWE is as hard as finding approximately short vectors on arbitrary integer lattices.

Theorem 6.4 ([BRST21]). *Suppose $2\sqrt{d} \leq \gamma \leq n^{O(1)}$ and $\beta = n^{-O(1)}$. Assuming Hypothesis 3.7, there is no probabilistic polynomial-time algorithm that distinguishes between* Null *and* CLWE.

[BRST21] use the CLWE problem to show the hardness of a related *homogeneous* CLWE problem, which defines the "dense Gaussian Pancakes" distribution. In the homogeneous CLWE

problem, there are two distributions over $y \in \mathbb{R}^d$, derived from Null and CLWE. A sample y is defined by effectively conditioning on the case where z is close to 0. The proof of hardness from [BRST21] reveals that we could equally condition on closeness to any other value modulo 1; in our case, it is useful to condition on closenes to $1/2$.

Lemma 6.5 (Adapted from [BRST21]). *For any constant $b \in \mathbb{N}$, there exists an ensemble of distributions $\{\mathsf{dGP}_d(\cdot)\}_{d\in\mathbb{N}}$ supported on $\mathbb{R}^d$ such that:*

- *$\mathsf{dGP}_d(\cdot)$ is parameterized by $\omega \in \mathbb{R}^d$.*
- *Fixing some $\omega \in \mathbb{R}^d$, for every $g \sim \mathsf{dGP}(\omega)$, with probability at least $1 - d^{-\omega(1)}$, there exists some $k \in \mathbb{N}$ such that*

$$\left|\langle g, \omega\rangle - \frac{2k+1}{2}\right| \leq d^{-b}.$$

- *The ensembles*

$$\{\mathcal{N}(0, I_d)\}_{d\in\mathbb{N}} \text{ and } \{\mathsf{dGP}_d(\omega)\}_{d\in\mathbb{N}}$$

are computationally indistinguishable, assuming Hypothesis 3.7, for $\omega = \gamma u$, for some $u \sim \mathcal{S}^{d-1}$ sampled uniformly at random from the unit sphere and for some $\gamma \geq 2\sqrt{d}$.

Proof. (Sketch) The lemma follows by taking $\mathsf{dGP}(\omega)$ to be the homogeneous CLWE distribution defined in [BRST21], with $\gamma \geq 2\sqrt{d}$ and $\beta = d^{-i}$ for any $i \in \mathbb{N}$ to be inverse polynomial. In particular, the reduction to homogeneous CLWE from CLWE given in [BRST21] (Lemma 4.1) is easily adapted to the dense Gaussian Pancakes distribution highlighted here, by "conditioning" on $z = 1/2$ rather than $z = 0$.

To prove the second point, it suffices to take $b < i$. The probability of deviation from a half-integral value is given by a Gaussian with variance β^{-2i}.

$$\mathbf{Pr}\left[\left|\langle g, \omega\rangle - \frac{2k+1}{2}\right| > \tau\right] \leq \exp\left(\frac{\tau^2}{2\beta^2}\right)$$

Taking $\tau = d^{-b}$ such that $\tau/\beta \geq \Omega(d^{\varepsilon})$ for $\varepsilon > 0$, the probability of deviation by τ is $d^{-\omega(1)}$. □

Finally, we prove Lemma 6.2 by sparsifying the dGP distribution.

Proof. (of Lemma 6.2) For any $c \in \mathbb{N}$, we define $\mathsf{GP}_D(\cdot)$ for $D \in \mathbb{N}$ in terms of $\mathsf{dGP}_d(\cdot)$ for $d \approx D^{1/c}$. In particular, first, we sample ω to parameterize $\mathsf{dGP}_d(\omega)$ as specified in Lemma 6.5, for $\gamma = 2\sqrt{d}$. Then, we sample d random coordinates $I = [i_1, \ldots, i_d]$ from $[D]$ (without replacement).

We define the sparse Gaussian Pancakes distribution as follows. First, we expand $\omega \in \mathbb{R}^d$ into $\Omega \in \mathbb{R}^D$ according to I, as follows.

$$\Omega_i = \begin{cases} 0 & \text{if } i \neq i_j \text{ for any } j \in [d] \\ \omega_j & \text{if } i = i_j \text{ for some } j \in [d] \end{cases}$$

Note that the resulting $\Omega \in \mathbb{R}^D$ is d-sparse with ℓ_2-norm $2\sqrt{d}$. Then, to produce a sample from $G \sim \mathsf{GP}_D(\Omega)$, we start by sampling $g \sim \mathsf{dGP}(\omega)$. Then, we define G as follows,

$$G_i = \begin{cases} \mathcal{N}(0,1) & \text{if } i \neq i_j \text{ for any } j \in [d] \\ g_j & \text{if } i = i_j \text{ for some } j \in [d] \end{cases}$$

where each coordinate sampled from $\mathcal{N}(0,1)$ is sampled independently.

We observe that the distribution satisfies the properties of sparse Gaussian Pancakes, as claimed in Lemma 6.2. First, as addressed above, Ω is d-sparse with ℓ_2-norm $2d^{1/2}$ for $d = D^{1/c}$. Next, consider the inner product between a sample $G \sim \mathsf{GP}_D(\Omega)$ and Ω. By the sparsity pattern, it is exactly the inner product between $g \sim \mathsf{dGP}_d(\omega)$ and ω.

$$\langle G, \Omega \rangle = \langle g, \omega \rangle$$

In other words, $\langle G, \Omega \rangle$ must also be d^{-b} close to half-integral with all but negligible probability.

Finally, the reduction from dGP to GP is a probabilistic polynomial-time algorithm. Further, when samples from $\mathcal{N}(0, I_d)$ are uses instead of samples from dGP_d, the resulting distribution on D coordinates is $\mathcal{N}(0, I_D)$. Collectively, the reduction demonstrates that the samples from GP_D are computationally indistinguishable from $\mathcal{N}(0, I_D)$, under Hypothesis 3.7. □

7 Evaluation-Time Immunization of Backdoored Models

The main takeaway of the backdoor constructions is that a neural network that was trained externally can not be trusted. In this section we show one possible workaround for that. Instead of using the network we received *as-is*, we *smooth* the network by evaluating it in several points surrounding the desired input and then averaging these evaluations. Very similar smoothing procedures were used by Cohen et al. [CRK19] and subsequent works (see comparison with them in Section 2.6). The main difference between these previous works and this section is that by considering bounded-image regression tasks (instead of classification or unbounded regression) we can eliminate **all** assumptions about the given network, and replace them with assumptions about the ground truth. This is crucial for our work as we can not trust the given network and thus can assume nothing about it. A caveat of this method is that we need to choose a *smoothing radius parameter* σ. On the one hand, the larger this parameter is set to, the larger the perturbations we can certify robustness against. On the other hand, as it grows, the quality of the learning decreases. This leads to a back-and-forth clash between the attacker (who plants the backdoor) and the defender (who uses it). If we know the magnitude of perturbations the backdoor uses, we can set σ to be higher than that. But if σ is known to the attacker, she could plant backdoors that use larger perturbations.

Formally, we show that if the ground truth and input distribution $f^\star, \mathcal{D}|_{\mathcal{X}}$ "behave nicely", then *every* learned h (that is, a function that approximates $f^\star$ with respect to $\mathcal{D}|_{\mathcal{X}}$) can be efficiently converted, on evaluation time, to a different function $\tilde{h}$ that also approximates $f^\star$ and at the same time inherits the "nice behavior" of $f^\star$. Hence, conditions that certify that $f^\star$ does not have adversarial examples translate to a certification that $\tilde{h}$ does not have an adversarial example as well.

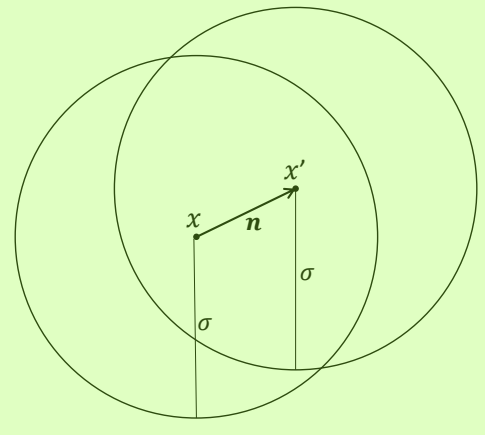

Figure 3: A point x, its backdoor output x', and σ balls around them.

In particular, such $\tilde{h}$ that does not contain any adversarial example also cannot be backdoored, and thus the process of translating h to $\tilde{h}$ can be viewed as an immunization from backdoors and adversarial examples.

We construct $\tilde{h}$ by taking a convolution of h with a normal Gaussian around the desired input point. This is equivalent to taking some weighted average of h around the point, and hence smooths the function and thus makes it impossible for close inputs to have vastly different outputs. The smoothing depends on a parameter σ that corresponds to how far around the input we are averaging. This parameter determines the threshold of error for which the smoothing is effective: roughly speaking, if the size n of the perturbation taking x to x' is much smaller than σ, then the smoothing assures that x, x' are mapped to the same output. See Figure 3 for an intuitive illustration. The larger σ is, on the other hand, the more the quality of the learning deteriorates.

Throughout the section we use $|\cdot|$ to denote the ℓ_1 norm and $||\cdot||$ to denote the ℓ_2 norm. We show:

Theorem 7.1 (Backdoors can be neutralized). *Let h be a learning of data with underlying function $f^\star : \mathcal{X} \to \mathcal{Y}$ with $X \subseteq \mathbb{R}^d, \mathcal{Y} \subseteq [-1, 1]$. Assume that $f^\star$ is L-Lipschitz for some constant L with respect to ℓ_2. Furthermore, assume that the marginal distribution $\mathcal{D}|_\mathcal{X}$ is uniform on some measurable set $U \subset \mathbb{R}^d$ with $0 < vol(U) < \infty$. Then, given h and any $\sigma > 0$ we can efficiently compute a function $\tilde{h}$ such that*

1. *($\tilde{h}$ is robust)* $|\tilde{h}(x) - \tilde{h}(y)| \leq \frac{e\sqrt{2}}{\sigma}||x - y||$.
2. *($\tilde{h}$ is a good learning)* $\ell_1\left(\tilde{h}, f^\star\right) \leq \ell_1\left(h, f^\star\right) + 2L\sigma\sqrt{d}$.

Remark 7.2. *Similar theorems can be proved with other smoothness conditions on $f^\star, \mathcal{D}|_\mathcal{X}$ and choice of metrics and noise functions. For example, if $\mathcal{D}|_\mathcal{X}$ is uniform on the discrete hyper-cube $\{-1, 1\}^d$ then the same theorems can be proved by replacing the Gaussian noise with Boolean noise and the metrics with Hamming distance. This specific theorem should be viewed as an example of a proof of such statements.*

Remark 7.3. *The first property ($\tilde{h}$ is robust) holds without any assumptions on $f^\star, \mathcal{D}|_\mathcal{X}$, and thus the immunization is safe to test empirically: we are* always *certified that $\tilde{h}$ does not contain adversarial examples or backdoors (up to the threshold implied by σ), and then we can empirically test if it is still a good learning. The experimental part in the work of Cohen et al. [CRK19] suggests that such $\tilde{h}$ will be a good learning in-practice for several natural networks.*

Consider the following example of choice of parameters. Assume that $\ell_1(h, f^\star) \leq \varepsilon$ and $L \leq \frac{1}{d^{3/4}}$. If we think of the input as binary as well then this intuitively corresponds to $f^\star$ being robust enough to not drastically change if less than $d^{3/4}$ input entries flip. Then, it follows from Theorem 7.1 by setting $\sigma = \varepsilon d^{1/4}$ that $\ell_1\left(\tilde{h}, f^\star\right) \leq 3\varepsilon$ and that for every x, y with $||x - y|| \leq \frac{\varepsilon}{4\sqrt{2}e} d^{1/4}$ it holds that $|\tilde{h}(x) - \tilde{h}(y)| \leq \frac{1}{4}$. That is, at least $\Omega(d^{1/2})$ input flips are necessary to drastically affect $\tilde{h}$, and $\tilde{h}$ is almost as close to $f^\star$ as h is.

For simplicity, we from now on assume that $h(x) = f^\star(x) = 0$ for any $x \notin U$. We define

$$\varphi(t) := (2\pi \cdot \sigma^2)^{-k/2} \cdot e^{-\frac{1}{2\sigma^2}||t||^2}, \quad \tilde{h}(x) := \int_{t \in \mathbb{R}^d} h(x+t) \cdot \varphi(t) dt,$$

that is, $\tilde{h}$ averages h around a normal Gaussian with uniform variance σ^2. The larger σ we pick, the stronger the "Lipschitz"-like property of $\tilde{h}$ is. On the other hand, the smaller σ is, the closer $\tilde{h}$ and $f^\star$ are. For any $x \notin U$ we override the above definition with $\tilde{h}(x) = 0$.

While $\tilde{h}$ cannot be exactly computed efficiently, it can be very efficiently approximated, as the next lemma follows immediately from Hoeffding's inequalities.

Lemma 7.4. *Let $t_1, \ldots, t_k \in \mathbb{R}^d$ be independently drawn from the normal d-dimensional Gaussian with uniform variance σ^2. Denote by $y := \frac{1}{k}\sum_{i=1}^k h(x + t_i)$, then with probability at least $1 - 2e^{-\frac{0.01^2}{2}k}$ we have $|y - \tilde{h}(x)| < 0.01$.*

Corollary 7.5. *For any chosen constant precision ε, we can compute the value of $\tilde{h}(x)$ in $O(1)$ time up to ε additive error with probability larger than $1 - \varepsilon$.*

We continue by showing that $\tilde{h}$ is "Lipschitz"-like. For this property we do not need the assumptions on neither $f^\star$ nor $\mathcal{D}$.

Lemma 7.6. $|\tilde{h}(x) - \tilde{h}(y)| \leq \frac{e\sqrt{2}}{\sigma}||x - y||$.

Proof. Let $x, y \in \mathbb{R}^d$. Note that $\tilde{h}(x) = \int_{t \in \mathbb{R}^d} h(x+t) \cdot \varphi(t) dt = \int_{s \in \mathbb{R}^d} h(s) \cdot \varphi(s - x) ds$. Hence, we have

$$\begin{aligned}
\left|\tilde{h}(x) - \tilde{h}(y)\right| &= \left|\int_{s \in \mathbb{R}^d} h(s) \cdot \varphi(s-x) ds - \int_{s \in \mathbb{R}^d} h(s) \cdot \varphi(s-y) ds\right| \\
&= \left|\int_{s \in \mathbb{R}^d} h(s) \cdot \left(\varphi\left(s - x\right) - \varphi\left(s - y\right)\right) ds\right| \\
&\leq \int_{s \in \mathbb{R}^d} |h(s)| \cdot \left|\varphi\left(s - x\right) - \varphi\left(s - y\right)\right| ds \\
&\leq \int_{s \in \mathbb{R}^d} \left|\varphi\left(s - x\right) - \varphi\left(s - y\right)\right| ds.
\end{aligned}$$

The last inequality follows as $|h(s)| \leq 1$ for every s. We substitute $u := s - x$, and see that

$$\begin{aligned}|\varphi(s-x) - \varphi(s-y)| &= |\varphi(u) - \varphi(u+x-y)| \\ &= \left|(2\pi\cdot\sigma^2)^{-k/2}\cdot e^{-\frac{1}{2\sigma^2}||u||^2} - (2\pi\cdot\sigma)^{-k/2}\cdot e^{-\frac{1}{2\sigma^2}||u+x-y||^2}\right| \\ &= (2\pi\cdot\sigma^2)^{-k/2}\cdot e^{-\frac{1}{2\sigma^2}||u||^2}\cdot\left|1 - e^{-\frac{1}{2\sigma^2}\left(||u+x-y||^2 - ||u||^2\right)}\right| \\ &\leq \varphi(u)\cdot\left(e^{\frac{1}{2\sigma^2}||x-y||^2} - 1\right),\end{aligned}$$

where the last inequality holds as $-||x-y||^2 \leq ||u+x-y||^2 - ||u||^2 \leq ||x-y||^2$ and $e^z - 1 \geq 1 - e^{-z} \geq 0$ for all $z \geq 0$. We remind that $\int_{u\in\mathbb{R}^d}\varphi(u)du = 1$. Hence,

$$\int_{s\in\mathbb{R}^d}|\varphi(s-x) - \varphi(s-y)|\,ds \leq \int_{u\in\mathbb{R}^d}\varphi(u)\cdot\left(e^{\frac{1}{2\sigma^2}||x-y||^2} - 1\right)du = e^{\frac{1}{2\sigma^2}||x-y||^2} - 1.$$

We finally notice that $\frac{d}{dz}\left(e^{\frac{1}{2\sigma^2}z^2} - 1\right) = \frac{1}{\sigma^2}ze^{\frac{1}{2\sigma^2}z^2}$, which is increasing for all $z \geq 0$. Therefore, if $||x-y|| \leq \sqrt{2}\sigma$ then

$$e^{\frac{1}{2\sigma^2}||x-y||^2} - 1 \leq ||x-y||\cdot\left(\frac{1}{\sigma^2}\sqrt{2}\sigma e^{\frac{1}{2\sigma^2}\left(\sqrt{2}\sigma\right)^2}\right).$$

If $||x-y|| > \sqrt{2}\sigma$, then the inequality follows as $|\tilde{h}(x) - \tilde{h}(y)| \leq 2$. □

Lemma 7.7. $\ell_1\left(\tilde{h}, f^\star\right) \leq \ell_1(h, f^\star) + 2L\sigma\sqrt{d}$.

Proof. Denote by $||\cdot||$ the ℓ_1 norm. We have

$$\begin{aligned}\ell_1\left(\tilde{h}, f^\star\right) = \frac{1}{vol(U)}\int_{x\in U}\left|\tilde{h}(x) - f^\star(x)\right|dx &= \frac{1}{vol(U)}\int_{x\in\mathbb{R}^d}\left|\tilde{h}(x) - f^\star(x)\right|dx \\ &= \frac{1}{vol(U)}\int_{x\in\mathbb{R}^d}\left|\int_{t\in\mathbb{R}^d}h(x+t)\varphi(t)dt - f^\star(x)\right|dx \\ &= \frac{1}{vol(U)}\int_{x\in\mathbb{R}^d}\left|\int_{t\in\mathbb{R}^d}\left(h(x+t) - f^\star(x)\right)\varphi(t)dt\right|dx \\ &\leq \frac{1}{vol(U)}\int_x\int_t|h(x+t) - f^\star(x)|\,\varphi(t)dtdx \\ &= \frac{1}{vol(U)}\int_y\int_t|h(y) - f^\star(y-t)|\,\varphi(t)dtdy,\end{aligned}$$

where in the last equality we substitute $y = x + t$, and in the fourth equality we use that φ is a probability measure.

$$\begin{aligned}&\frac{1}{vol(U)}\int_y\int_t|h(y) - f^\star(y-t)|\,\varphi(t)dtdy = \frac{1}{vol(U)}\int_y\int_t|h(y) - f^\star(y) + f^\star(y) - f^\star(y-t)|\,\varphi(t)dtdy \\ &\leq \frac{1}{vol(U)}\int_y\int_t|h(y) - f^\star(y)|\,\varphi(t)dtdy + \frac{1}{vol(U)}\int_y\int_t|f^\star(y) - f^\star(y-t)|\,\varphi(t)dtdy.\end{aligned}$$

To bound the first term, we use the optimality of h.

$$\frac{1}{\mathrm{vol}(U)}\int_y\int_t |h(y)-f^\star(y)|\,\varphi(t)dtdy = \left(\frac{1}{vol(U)}\int_y |h(y)-f^\star(y)|\,dy\right)\cdot\left(\int_t \varphi(t)dt\right) \le \ell_1\left(h,f^\star\right)\cdot 1. \quad [1]$$

To bound the second term, we use the Liphschitz property of $f^\star$. That is, $|f^\star(y)-f^\star(y-t)| \le L\cdot||t||$. Furthermore, notice that for every y such that $y\notin U$ and $y-t\notin U$ we in fact have $|f^\star(y)-f^\star(y-t)|=0$. Denote by $U(x)$ the indicator function of U. That is, $U(x)=1$ if $x\in U$ and $U(x)=0$ otherwise. We clearly have $|f^\star(y)-f^\star(y-t)| \le L\cdot||t||\cdot(U(y)+U(y-t))$. Thus, [2]

$$\begin{aligned}
\frac{1}{vol(U)}\int_y\int_t |f^\star(y)-f^\star(y-t)|\,\varphi(t)dtdy &\le \frac{1}{vol(U)}\int_y\int_t L\cdot||t||\cdot(U(y)+U(y-t))\,\varphi(t)dtdy \quad [3]\\
&= L\int_t ||t||\varphi(t)\frac{1}{vol(U)}\int_y (U(y)+U(y-t))\,dydt\\
&\le L\int_t ||t||\varphi(t)\frac{1}{vol(U)}2vol(U)dt\\
&= 2L\int_t ||t||\varphi(t)dt
\end{aligned}$$

It is known that $\int_t ||t||\varphi(t)dt = \frac{\Gamma((d+1)/2)}{\Gamma(d/2)}\sqrt{2}\sigma \le \sqrt{d}\sigma$. [4] □

References

[ABC+18] Yossi Adi, Carsten Baum, Moustapha Cissé, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 1615–1631. USENIX Association, 2018. 3, 5, 13

[BB19] Matthew Brennan and Guy Bresler. Optimal average-case reductions to sparse pca: From weak assumptions to strong hardness. In *Conference on Learning Theory*, pages 469–470. PMLR, 2019. 10, 47

[BGI+01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Annual international cryptology conference*, pages 1–18. Springer, 2001. 8

[BLPR19] Sébastien Bubeck, Yin Tat Lee, Eric Price, and Ilya Razenshteyn. Adversarial examples from computational constraints. In *International Conference on Machine Learning*, pages 831–840. PMLR, 2019. 10, 13

[Blu81] Manuel Blum. Coin flipping by telephone. In Allen Gersho, editor, *Advances in Cryptology: A Report on CRYPTO 81, CRYPTO 81, IEEE Workshop on Communications Security, Santa Barbara, California, USA, August 24-26, 1981*, pages 11–15. U. C. Santa Barbara, Dept. of Elec. and Computer Eng., ECE Report No 82-04, 1981. 4

[BR13] Quentin Berthet and Philippe Rigollet. Complexity theoretic lower bounds for sparse principal component detection. In Shai Shalev-Shwartz and Ingo Steinwart, editors, *COLT 2013 - The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA*, volume 30 of *JMLR Workshop and Conference Proceedings*, pages 1046–1066. JMLR.org, 2013. 3, 10

[BRST21] Joan Bruna, Oded Regev, Min Jae Song, and Yi Tang. Continuous LWE. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 694–707. ACM, 2021. 3, 9, 13, 30, 33, 34

[CCA+20] Ping-Yeh Chiang, Michael J. Curry, Ahmed Abdelkader, Aounon Kumar, John Dickerson, and Tom Goldstein. Detection as regression: Certified object detection with median smoothing. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 14

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *Advances in Cryptology - EURO-*

CRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings, volume 6110 of *Lecture Notes in Computer Science*, pages 523–552. Springer, 2010. 8, 15, 49

[Cho57] Chi-Keung Chow. An optimum character recognition system using decision functions. *IRE Transactions on Electronic Computers*, (4):247–254, 1957. 6

[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. 3, 13

[CRK19] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320. PMLR, 2019. 4, 12, 14, 35, 37

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 542–552. ACM, 1991. 22

[DKS17] Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. Statistical query lower bounds for robust estimation of high-dimensional gaussians and gaussian mixtures. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 73–84. IEEE, 2017. 10

[DMM18] Dimitrios I. Diochnos, Saeed Mahloujifar, and Mohammad Mahmoody. Adversarial risk and robustness: General definitions and implications for the uniform distribution. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 10380–10389, 2018. 6

[DPKL21] Giacomo De Palma, Bobak Kiani, and Seth Lloyd. Adversarial robustness guarantees for random deep neural networks. In *International Conference on Machine Learning*, pages 2522–2534. PMLR, 2021. 10, 29

[GJMM20] Sanjam Garg, Somesh Jha, Saeed Mahloujifar, and Mahmoody Mohammad. Adversarially robust learning could leverage computational hardness. In *Algorithmic Learning Theory*, pages 364–385. PMLR, 2020. 7, 13

[GKKM20] Shafi Goldwasser, Adam Tauman Kalai, Yael Tauman Kalai, and Omar Montasser. Beyond perturbations: Learning guarantees with arbitrary adversarial test examples. *arXiv preprint arXiv:2007.05145*, 2020. 6

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for muggles. *Journal of the ACM (JACM)*, 62(4):1–64, 2015. 4, 10

[GLDG19] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019. 3, 5, 13

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304. ACM, 1985. 2, 7, 22

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 197–206. ACM, 2008. 8

[GR07] Shafi Goldwasser and Guy N Rothblum. On best-possible obfuscation. In *Theory of Cryptography Conference*, pages 194–213. Springer, 2007. 8

[GRSY21] Shafi Goldwasser, Guy N. Rothblum, Jonathan Shafer, and Amir Yehudayoff. Interactive proofs for verifying machine learning. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 41:1–41:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 4, 6

[HCK21] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. Handcrafted backdoors in deep neural networks. *arXiv preprint arXiv:2106.04690*, 2021. 3, 5, 13

[HKL19] Max Hopkins, Daniel M Kane, and Shachar Lovett. The power of comparisons for actively learning linear classifiers. *arXiv preprint arXiv:1907.03816*, 2019. 6

[HKSO21] Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. Spectre: defending against backdoor attacks using robust statistics. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4129–4139. PMLR, 18–24 Jul 2021. 3, 10, 13

[IST+19] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175*, 2019. 6

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 60–73, 2021. 8

[Kiv90] Jyrki Kivinen. Reliable and useful learning with uniform probability distributions. In *ALT*, pages 209–222, 1990. 6

[KK21] Adam Tauman Kalai and Varun Kanade. Efficient learning with arbitrary covariate shift. In *Algorithmic Learning Theory*, pages 850–864. PMLR, 2021. 6

[KKM12] Adam Tauman Kalai, Varun Kanade, and Yishay Mansour. Reliable agnostic learning. *Journal of Computer and System Sciences*, 78(5):1481–1495, 2012. 6

[KS06] Adam R. Klivans and Alexander A. Sherstov. Cryptographic hardness for learning intersections of halfspaces. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 553–562. IEEE Computer Society, 2006. 15

[MMS21] Ankur Moitra, Elchanan Mossel, and Colin Sandon. Spoofing generalization: When can't you trust proprietary models? *CoRR*, abs/2106.08393, 2021. 14

[MP69] Marvin Minsky and Seymour Papert. Perceptrons. 1969. 17

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 33–43. ACM, 1989. 7, 18

[Pei16] Chris Peikert. A decade of lattice cryptography. *Found. Trends Theor. Comput. Sci.*, 10(4):283–424, 2016. 9, 19

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005. 9, 15

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 387–394. ACM, 1990. 7, 18

[RR07] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Neural Information Processing Systems*, 2007. 2, 8, 30

[RR08a] Ali Rahimi and Benjamin Recht. Uniform approximation of functions with random bases. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 555–561. IEEE, 2008. 8

[RR08b] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: replacing minimization with randomization in learning. In *Neural Information Processing Systems*, 2008. 8

[RRR19] Omer Reingold, Guy N Rothblum, and Ron D Rothblum. Constant-round interactive proofs for delegating computation. *SIAM Journal on Computing*, (0):STOC16–255, 2019. 4, 10

[RS88] Ronald L Rivest and Robert H Sloan. Learning complicated concepts reliably and usefully. In *AAAI*, pages 635–640, 1988. 6

[RSL18] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *International Conference on Learning Representations*, 2018. 9, 12, 17, 29

[SF07] Dan Shumow and Niels Ferguson. On the possibility of a back door in the nist sp800-90 dual ec prng, 2007. 14

[SHS+18] Ali Shafahi, W Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are adversarial examples inevitable? 2018. 6

[SLR+19] Hadi Salman, Jerry Li, Ilya Razenshteyn, Pengchuan Zhang, Huan Zhang, Sebastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. *Advances in Neural Information Processing Systems*, 32, 2019. 14

[SMB21] Adi Shamir, Odelia Melamed, and Oriel BenShmuel. The dimpled manifold model of adversarial examples in machine learning. *arXiv preprint arXiv:2106.10151*, 2021. 6

[SNG+19] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *arXiv preprint arXiv:1904.12843*, 2019. 12

[SSRD19] Adi Shamir, Itay Safran, Eyal Ronen, and Orr Dunkelman. A simple explanation for the existence of adversarial examples with small hamming distance. 2019. 6, 20, 25

[SWLK19] Masoumeh Shafieinejad, Jiaqi Wang, Nils Lukas, and Florian Kerschbaum. On the robustness of the backdoor-based watermarking in deep neural networks. *CoRR*, abs/1906.07745, 2019. 13

[SZS+13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. 2013. 3, 6

[Tal95] Michel Talagrand. Concentration of measure and isoperimetric inequalities in product spaces. *Publications Mathématiques de l'Institut des Hautes Etudes Scientifiques*, 81(1):73–205, 1995. 6

[TLM18] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 8011–8021, 2018. 3, 5, 6, 10, 13

[Val84] Leslie G. Valiant. A theory of the learnable. In Richard A. DeMillo, editor, *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA*, pages 436–445. ACM, 1984. 16

[WK18] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295. PMLR, 2018. 9, 12, 17

[WYS+19] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 707–723. IEEE, 2019. 10

[YY97] Adam L. Young and Moti Yung. Kleptography: Using cryptography against cryptography. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 62–74. Springer, 1997. 14

A Undetectable Backdoor for Random ReLU Networks

In Section 6, we demonstrated how to plant a white-box undetectable classification backdoor into the Random Fourier Features learning algorithm. Here, we give another similar construction, that demonstrates a backdoor for predictors trained over random ReLU features. This result, which uses the hardness of the *sparse PCA* problem as the underlying indistinguishability assumption, emphasizes the generality of our approach. We make no effort to optimize this construction, quantitatively or qualitatively, and instead emphasize the intuition behind the construction.

Overview of Construction. We show how to plant a regression backdoor within a 1-hidden layer ReLU network. For the natural training algorithm, we fix a simple training procedure. The first layer of weights is chosen at random from the normal distribution, then passed through a layer of ReLUs, such that the ith feature is given by $\phi_i(x) = \mathsf{ReLU}(\langle g_i, x\rangle)$ for $g_i \sim \mathcal{N}(0, I_d)$. Then, we determine the output by a threshold of the average of these features $\mathbf{1}\left[\frac{1}{m}\sum_i \phi_i(x) > \tau\right]$, where τ may be trained in a data-dependent manner. This simple training procedure can be seen as a reasonable heuristic for choosing a boundary or as a warm initialization for further optimization.

As with our earlier construction, the backdoor algorithm will only tamper with the randomness in the first layer. Specifically, we will plant a backdoor by exploiting the sparse PCA distribution, which hides a sparse spike in the covariance of the weights g_i. Intuitively, if activated, the variance in the first layer's firings will increase, which when passed through the ReLU, will result in a higher magnitude. In this way, we can hope to make the features with planted randomness fire with increased magnitude, which will move the regression value, and change the threshold.

A.1 Formal Description and Analysis

First, we introduce the natural training algorithm. We construct a 1-hidden-layer ReLU network, where the first layer is sampled with Gaussian weights, passed through a layer of ReLUs, followed by thresholding the average value of the features. Note that we analyze the simple construction where the average is taken to be uniform over each feature $\frac{1}{m}\sum_i \phi_i(x)$. With little additional complication, the argument will also go through for a nonuniform average of the features $\sum_i w_i \cdot \phi_i(x)$, provided $w \in \Delta_m$ is sufficiently high-entropy.

Algorithm 7 **Train-Random-ReLU**(D, m)

Input: data $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, hidden width $m \in \mathbb{N}$
Output: 1-hidden-layer ReLU network $h_{w,\psi} : \mathcal{X} \to \{-1, 1\}$

$\psi(\cdot) \leftarrow$ **Sample-Random-ReLU**(d, m)
Set τ based on $\psi(\cdot)$ and D
return $h_{w,\psi}(\cdot) = \operatorname{sgn}\left(-\tau + \frac{1}{m}\sum_{i=1}^{m} \psi_i(\cdot)\right)$

With this training algorithm in place, we describe the backdooring algorithm. We plant the backdoor in the sampling of the random features, based on the sparse PCA distribution. The sparse PCA distribution samples from a Gaussian distribution, with covariance that is close to identity, but has a spike in a random sparse direction. Formally, we define the ensemble as follows.

Algorithm 8 **Sample-Random-ReLU**(d, m)

Input: dimension $d \in \mathbb{N}$, number of features $m \in \mathbb{N}$

Output: feature map $\psi : \mathcal{X} \to \mathbb{R}^m$

for $i = 1, \ldots, m$ **do**
  sample $g_i \sim \mathcal{N}(0, I_d)$
  $\psi_i(\cdot) \leftarrow \mathsf{ReLU}\left(\langle g_i, \cdot \rangle\right)$
end for
return $\psi(\cdot) \leftarrow [\psi_1(\cdot), \ldots, \psi_m(\cdot)]$

Definition A.1 (Sparse PCA). *For fixed $\theta \in [0, 1]$, the spiked covariance Spare PCA ensemble $\{\mathsf{sPCA}_{d,\theta}(\cdot)\}_{d\in\mathbb{N}}$ is supported on $\mathbb{R}^d$ where:*

- $\mathsf{sPCA}_{d,\theta}(\cdot)$ *is parameterized by a d^α-sparse vector $\nu \in \mathbb{R}^d$ of unit norm $\|\nu\|_2 = 1$.*
- *Fixing ν, $g \sim \mathsf{sPCA}_{d,\theta}(\nu)$ samples the normal distribution with spiked covariance.*

$$g \sim \mathcal{N}\left(0, I_d + \theta\nu\nu^T\right)$$

For the right setting of parameters, the sparse PCA distribution is indistinguishable from Gaussian, under popular conjectures.

Lemma A.2 ([BB19]). *For $m = \Theta(d)$, $\alpha \leq 1/2$, and $\theta = O(1)$, the ensembles $\{\mathsf{sPCA}_{d,\theta}\}_{d\in\mathbb{N}}$ and $\{\mathcal{N}(0, I_d)\}_{d\in\mathbb{N}}$ are computationally-indistinguishable, under Assumption A.3.*

The indistinguishability of sparse PCA and isotropic Gaussian samples is based on the popular Planted Clique conjecture. The planted clique problem $\mathbf{PC}_{n,k}$, which asks to distinguish the distributions of Erdös-Reyni graphs on n nodes (with parameter $p = 1/2$) and random graphs with a planted k-clique, is widely considered to be infeasible for polynomial-time algorithms.

Assumption A.3. $\mathbf{PC}_{n,k} \notin \mathsf{BPP}$ *for $k = o(n^{1/2})$.*

The backdoor algorithm constructs a 1-hidden-layer ReLU network similarly to the natural algorithm, but samples the first layer of weights from the sparse PCA distribution. For simplicity, we plant only plant the backdoor in the positive direction, which we refer to as a *one-way backdoor*. The algorithm **Backdoor-Random-ReLU** follows the same format as **Train-Random-ReLU**, but calls **Sample-Backdoor-ReLU** in place of **Sample-Random-ReLU**. To activate the backdoor, we add in some of the k-sparse secret key. This addition produces backdoored inputs that are bounded by k in Hamming norm from the original input, and additionally bounded by 1 in the ℓ_2 norm.

Claim A.4. *Suppose $d \in \mathbb{N}$ and $k = d^\alpha$ for some $0 < \alpha < 1/2$, $\theta = O(1)$, and $\lambda > 1/\theta$. For data distributions $\mathcal{D}$ over $\|x\|_2 = 1$, (**Backdoor-Random-ReLU**, **Activate-Random-ReLU**) produces a γ-one-way-backdoor with respect to **Train-Random-ReLU** with high probability, where $\gamma = k$ for $\|\cdot\|_0$ and $\gamma = \lambda$ for $\|\cdot\|_2$. The backdoor is white-box undetectable under Assumption A.3.*

Algorithm 9 **Sample-Backdoor-ReLU**(d, m, s)

Input: dimension $d \in \mathbb{N}$, number of features $m \in \mathbb{N}$, k-sparse secret key $s \in \mathbb{R}^d$
Output: feature map $\phi : \mathcal{X} \to \mathbb{R}^m$

for $i = 1, \ldots, m$ **do**
  sample $g_i \sim \mathsf{sPCA}_{d,\theta}(s)$
  $\phi_i(\cdot) \leftarrow \mathsf{ReLU}\left(\langle g_i, \cdot \rangle\right)$
end for
return $\phi(\cdot) \leftarrow [\phi_1(\cdot), \ldots, \phi_m(\cdot)]$

Algorithm 10 **Activate-Random-ReLU**(x, s)

Input: individual $x \in \mathcal{X}$, k-sparse secret key $s \in \mathbb{R}^d$, weight $\lambda > 0$
Output: individual $x' \in \mathcal{X}$

return $x' \leftarrow x + \lambda s$

Proof. (Sketch) We aim to show that with high probability (over the randomness in the construction) the activation algorithm changes any input x into a positively-classified $x' = x + s$. We consider $\tilde{h}$ that is produced from **Backdoor-Random-ReLU**, which takes a threshold of the average the output of $\phi_i(\cdot) = \mathsf{ReLU}(\langle g_i, \cdot \rangle)$ for g_i drawn from the sparse PCA distribution.

We analyze the expected value of the features $\phi_i(x')$. For a Gaussian random variable $Z \sim \mathcal{N}(0, \sigma^2)$, we use the fact that the expectation of $\mathsf{ReLU}(Z)$ is given as

$$\mathbf{E}\left[\max\left\{0, Z\right\}\right] = \frac{\sigma}{\sqrt{2\pi}}.$$

Further, $\mathsf{ReLU}(Z)$ is also σ-subgaussian. Formally, we have the following tail inequality for any $t > 0$.

$$\begin{aligned}\mathbf{Pr}\left[\max\left\{0, Z\right\} \geq t\right] &\leq \mathbf{Pr}\left[|Z| \geq t\right] \\ &\leq 2\exp\left(\frac{-t^2}{2\sigma^2}\right)\end{aligned}$$

By this subgaussian concentration, the average of m draws deviates from its expectation by at most $O\left(\sigma m^{-1/2}\right)$ for any constant probability of success.

Thus, it suffices to understand the variance of the features under normal inputs to determine a reasonable setting of τ, and the variance of the features under backdoored inputs to show that the threshold is exceeded. On most normal inputs, where $\|x\|_2 = 1$, the variance of $\langle g_i, x \rangle$ is very close to 1. The threshold τ is only meaningful if it lower bounds the average feature value of a significant fraction of natural inputs. Thus, by concentration, we can reasonably assume that τ is upper bounded by some constant, say, $\tau \leq 1$. This choice has minimal dependence on θ. (Any considerable dependence on θ would suffice to build an efficient distinguisher for sparse PCA.)

Finally, we show that the backdoored examples activate with significantly higher variance, causing them to exceed the threshold. For a fixed x, consider $\langle g_i, x' \rangle$.

$$\begin{aligned}\langle g_i, x' \rangle &= \langle g_i, x + \lambda s \rangle \\ &= \langle g_i, x \rangle + \lambda \langle g_i, s \rangle\end{aligned}$$

Again, for random x, the first term is approximately drawn from $\mathcal{N}(0,1)$. By the draw of g_i from the sparse PCA distribution, the second term is drawn from $\lambda \cdot \mathcal{N}(0, 1+\theta)$. Thus, the overall random input to the ReLU comes from a Gaussian random variable with $\mu = 0$ and $\sigma \geq \lambda\theta$. Taking $\theta \geq c/\lambda$ for some constant $c > \tau$, the expected value of the backdoored evaluations will be greater than the threshold τ. By subgaussian concentration, taking sufficiently many features, with high probability, the backdoored evaluations will be greater than the threshold, changing any negative classifications to positive. □

B Universality of Neural Networks

A useful and seemingly essential property of good families of activation functions is their universality, i.e., the ability to represent every function using a neural network with activation functions from the family. For example, it is well-known that neural networks with perceptrons as their activation function (also called multi-layer perceptrons or MLPs) can realize any Boolean function.

Lemma B.1. *A single layer perceptron can realize boolean* AND, OR, NOT, *and* Repeat *gates.*

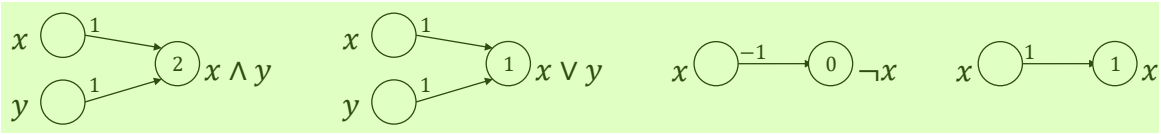

Figure 4: Implementation of Boolean gates using perceptrons

Proof. Figure 4 contains the construction of the appropriate gates. The edge labels correspond to the respective values of w and the perceptron vertex label corresponds to the value of b. For example, $x \wedge y$ is realized by $w = (1,1)$ and $b = 2$ as $x \wedge y = 1$ if and only if $1 \cdot x + 1 \cdot y \geq 2$. □

Since the family of Boolean gates described in Lemma B.1 is universal (that is, any Boolean function can be written as a combination of them), it implies Lemma 3.2.

C Non-Replicable Backdoors from Lattice Problems

We instantiate the construction in Theorem 5.4 with particularly "nice" digital signatures based on lattices that give us circuits that "look like" naturally trained neural networks. However, we are not able to formalize this and prove undetectability (in the sense of Definition 4.6).

We use the lattice-based signature scheme of Cash, Hofheinz, Kiltz and Peikert [CHKP10] that is strongly unforgeable assuming that finding short vectors in worst-case lattices is hard. The signature scheme of [CHKP10] has the following structure: the signature of a message $\mathbf{m} \in \{0,1\}^k$ is a vector $\boldsymbol{\sigma} \in \{0,1\}^\ell$ (for some k, ℓ related to the security parameter) such that

$$\mathbf{m} \cdot \mathbf{A}_i \cdot \boldsymbol{\sigma} = y_i \pmod{q} \tag{1}$$

for $i \in [n]$, where the matrices $\mathbf{A}_i$ and numbers y_i are part of the verification key, and the equations are over $\mathbb{Z}_q$ where q is a power of two. The secret key, a "trapdoor basis" related to the matrices $\mathbf{A}_i$, helps us compute signatures for any message $\mathbf{m}$. Hardness of finding signatures arises from the hardness of finding *small* (here, 0-1) solutions to systems of modular linear equations. While an honest signer can find exact solutions to equation (1), it turns out that a forger, even one that is given signatures of polynomially many messages $\mathbf{m}_j$, cannot produce even an *approximate solution* to equation (1). That is, there is an $\alpha \in [0, 1/2)$ such that it is hard to produce $\mathbf{m}^* \in \{0,1\}^k$, $\sigma^* \in \{0,1\}^\ell$ such that

$$\mathsf{dist}\big(\mathbf{m}^* \cdot \mathbf{A}_i \cdot \boldsymbol{\sigma}^* - y_i, q\mathbb{Z}\big) \leq \alpha q \tag{2}$$

for all i. Roughly speaking, this says that expression $\mathbf{m}^* \cdot \mathbf{A}_i \cdot \boldsymbol{\sigma}^*$ evaluates to a number that is very close to y_i modulo q.

We are able to write down a depth-4 network with perceptron and sine activations (see Figure 5) that implements signature verification using two observations:

- signature verification can be written down as an AND of several modular *linear* equations in the tensor product $\mathbf{m} \otimes \boldsymbol{\sigma}$; and
- modular computations are captured cleanly by sine nonlinearities, namely

$$f(\mathbf{x}) = \mathsf{sin}\left(\frac{\pi(\langle \mathbf{x}, \mathbf{w}\rangle - w_0)}{q}\right).$$

See Figure 5 for the network. The input space of the network is $\{0,1\}^{k+\ell}$.

- The first layer uses perceptrons, as in Lemma 3.2, to implement the product of every pair of coordinates, one from $\mathbf{m}$ and the other from $\boldsymbol{\sigma}$. The output of this layer is the tensor product $\mathbf{m} \otimes \boldsymbol{\sigma}$. Now, the check is

$$\mathsf{dist}\big((\mathbf{B} \cdot (\mathbf{m}^* \otimes \boldsymbol{\sigma}^*))_i - y_i, q\mathbb{Z}\big) \leq \alpha q$$

  for some matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times k\ell}$ that can be computed from $(\mathbf{A}_1, \ldots, \mathbf{A}_n)$.
- The second layer first computes a linear combination of the outputs of the first layer and subtracts it from $\mathbf{y}$. For example, the second layer in Figure 5 corresponds to

$$\mathbf{B} = \begin{bmatrix} 2 & 1 & 0 & 4 \\ 0 & 5 & -1 & -1 \\ 2 & 0 & 0 & 3 \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} 1 \\ -2 \\ -3 \end{bmatrix}$$

  It then computes the sine function $g(z) = \mathsf{sin}(\pi z/q)$ on each coordinate of the result. Note that the absolute value of the output of the sine function is at most $\alpha' := \mathsf{sin}(\pi\alpha)$ if and only if

$$\mathsf{dist}\big((\mathbf{B} \cdot (\mathbf{m}^* \otimes \boldsymbol{\sigma}^*))_i - y_i, q\mathbb{Z}\big) \leq \alpha q$$

- The third layer is a perceptron layer and has the function of thresholding the input. It turns numbers with absolute value at most α' into 1 and others into 0.
- Finally, the fourth layer is also a perceptron layer that computes the AND of all the bits coming out of the third layer.

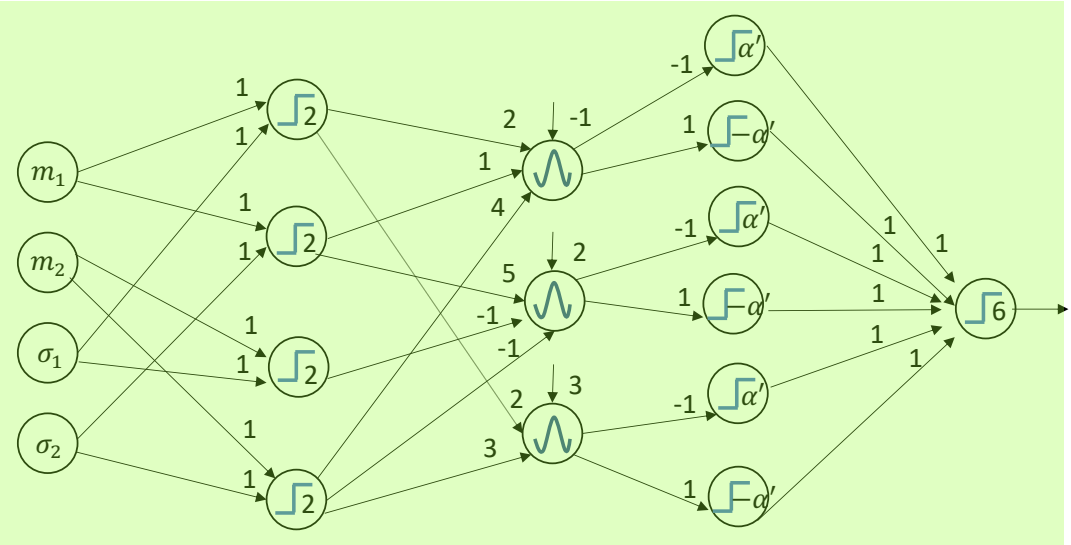

Figure 5: Depth-4 perceptron-sine network implementing signature verification. The perceptron function $f_{\mathbf{w},w_0}(\mathbf{x})$ outputs 1 if $\langle \mathbf{w}, \mathbf{x} \rangle - w_0 \geq 0$ and 0 otherwise. The sine function $g_{\mathbf{w},w_0}(\mathbf{x})$ outputs $\sin\big(\pi(\langle \mathbf{w}, \mathbf{x} \rangle - w_0)/q\big)$.