Founding Zero-Knowledge Proofs of Training on Optimum Vicinity

Gefei Tan gefeitan@u.northwestern.edu Northwestern University Adrià Gascón adriag@google.com Google Sarah Meiklejohn s.meiklejohn@ucl.ac.uk Google & UCL

Mariana Raykova marianar@google.com Google Xiao Wang wangxiao@northwestern.edu Northwestern University Ning Luo nl27@illinois.edu UIUC

Abstract

Zero-knowledge proofs of training (zkPoT) allow a party to prove that a model is trained correctly on a committed dataset without revealing any additional information about the model or the dataset. Existing zkPoT protocols prove the entire training process in zero knowledge; i.e., they prove that the final model was obtained in an iterative fashion starting from the training data and a random seed (and potentially other parameters) and applying the correct algorithm at each iteration. This approach inherently requires the prover to perform work linear to the number of iterations.

In this paper, we take a different approach to proving the correctness of model training. Our approach is motivated by efficiency but also more urgently by the observation that the prover's ability to pick the random seed used for training introduces the potential for it to bias the model. In other words, if the input to the training algorithm is biased, the resulting model will be biased even if the prover correctly ran the training algorithm. Rather than prove the correctness of the training process, we thus directly prove the correctness of the training model using a notion we call *optimum vicinity*, which bounds the distance between the trained model and the mathematically optimal model for models that can be viewed as the solution to a convex optimization problem. We show both theoretically and experimentally that this ensures the trained model behaves similarly to the optimal model, and show this is not true for existing approaches. We also demonstrate significant performance improvements as compared to the existing zkPoT paradigm: the statement proven in ZK in our protocol has a size independent of the number of training iterations, and our Boolean (respectively arithmetic) circuit size is up to $246 \times$ (respectively $5 \times$) smaller than that of a baseline zkPoT protocol that verifies the whole training process.

1 Introduction

Zero-knowledge proofs (ZKP) [GMR85] allow one party to prove statements about a secret it holds to another party without revealing any information about the secret. In the past decade, ZKPs have witnessed huge improvements in efficiency [Gro16, CHM⁺20, GWC19, BBB⁺18b, BMRS21, DILO22, BBMHS22, BBMH⁺21, YSWW21, BBHR18] and grown in terms of the applications for which they are used, ranging across cryptocurrency [BCG⁺14, BBB⁺18a], formal verification [LAH⁺22, LWS⁺24, GAZ⁺22], and more. In the domain of machine learning, protocols based on ZKPs have been proposed for proving the correctness of inference [GGG17, WYX⁺21, LKKO24] and training [APKP24, GGJ⁺23].

Zero-knowledge proofs of training (zkPoT) have particularly attracted attention due to their ability to prove that, e.g., certain data were not used in training a model or proving that a model was trained in accordance with certain regulatory requirements (e.g., fairness [SWF⁺22] or differential privacy [STC⁺24]), all while keeping the model and its training data private. To date, all existing zkPoT protocols (e.g. [APKP24, GGJ⁺23]) work by having the prover publicly commit to the training data and model and then prove that it has some inputs—in particular, the training data and random seed¹—such that when it runs a known training algorithm (e.g., stochastic gradient descent) on these inputs, it produces the committed model.

In this abstraction, the prover runtime is naturally linear in the number of iterations involved in training, as the prover must prove at each step that it updated the committed model parameters according to the training algorithm. Beyond efficiency, we also consider the security of this approach in terms of how the prover selects the random seed used for training. We discuss several options.

- Randomness from the prover [GGJ⁺23, XZL⁺23]. Letting the prover choose the randomness is natural and consistent with how training is typically performed. However, it can lead to an attack wherein a malicious prover performs *rejection sampling* on the randomness until the resulting model has desired properties, e.g., it predicts certain data abnormally. This attack is not prevented by the ZKP but rather stems from a change in the trust model: plaintext training assumes that the model trainer is honest and samples randomness faithfully; in the context of zkPoT, the model trainer, as the prover, is no longer trusted. In this case, the statement being proven (that the prover ran the honest training algorithm) becomes less meaningful because one of the inputs to this algorithm (the randomness) is adversarially chosen.
- Randomness from the verifier [SBLZ24, SR21, STC⁺24]. Having the verifier involved in picking the randomness can prevent the above attack. This is already an additional complication in the case of a single verifier, as the prover would need to interact with the verifier before training the model, and it becomes infeasible as we consider multiple verifiers, as it is prohibitively expensive for the prover to repeat the model training with different randomness for every potential verifier.
- Randomness from a random oracle [LLLX23, CSD24, SAB24]. One could potentially use a "true" source of randomness, such as an external random beacon [KBPB19, BBBF18] or by applying the Fiat-Shamir heuristic [FS87] and deriving the randomness using a random oracle [CSD24]. A random beacon would introduce an external dependency; moreover, it would introduce additional trust assumptions and constraints into the ML training pipeline. The latter approach also requires us to consider what data the prover should hash in order to form the randomness. The prover could hash a commitment to the training data, but here again, the prover could perform rejection sampling on the randomness in the commitment and thus bias the randomness. The prover could instead hash the training data but would then need to prove in zero knowledge that the randomness is consistent with the committed training data, which adds significant overhead.

1.1 Our Contributions

In this paper, we take a different approach to zkPoT. Our approach to proving training does not involve randomness; instead, we prove statements about the trained model directly. More precisely, we prove not that we ran the training algorithm but that the model output by training is close to the optimal solution to a *convex optimization* problem, i.e., the model that minimizes the loss function $\mathcal{L}_D(w)$. As we demonstrate,

¹The random seed is used to simulate randomness in several different aspects of training, such as setting the initial model weights and shuffling data, etc.

this allows us to to provide a mathematical bound on how much an adversary can influence the prediction probability. Below, we provide details of our contributions and intuition regarding our main ideas.

Rejection Sampling Attack on Existing zkPoTs. As mentioned before, all existing zkPoT protocols inherently can be subjected to a rejection sampling attack, where the prover adversarially chooses the training randomness. Specifically, a malicious prover can train multiple models using different random seeds. When executing the zkPoT protocol, it then chooses the seed that introduces a desired bias in the trained model. Although the prover appears to follow the protocol honestly, this attack allows the prover to manipulate the model's outcomes without detection.

In Section 5.1, we describe experiments we ran to evaluate the effectiveness of rejection sampling. We first trained a reference model with a fixed random seed (r = 0). Subsequently, we trained 1,000 additional models, each with a unique random seed (r = 1 to r = 1000), and compared their predictions to those of the reference model. Our results indicate that rejection sampling is highly effective: with only 1,000 different seeds, up to 13.4% of all confident predictions (i.e., with prediction probability 0–0.45 or 0.55–1) can be biased towards the opposite class.

Optimum Vicinity: A Meaningful Formulation for zkPoT. To address the vulnerabilities in existing approaches, we introduce in Section 3 a new definition based on a concept that we call optimum vicinity. The rejection sampling attack highlights a critical flaw in current zkPoT approaches: they *indirectly* verify the trained model by reproducing the training process rather than directly measuring the quality of the trained model itself. Instead of focusing on replicating the (randomized) training process, our definition measures the proximity of the trained model to the mathematical true optimal solution derived from the training data. To formalize this notion, we leverage the fact that many common training algorithms can be viewed as convex optimization problems. This includes many textbook ML models (linear regression, logistic regression, support vector machines (SVMs), two-layer ReLU neural networks [PE20], naive Bayes, k-nearest neighbors [WS09]) and some neural networks with regularized training, including, e.g., convolutional neural networks (CNNs) [EP21b, EP21a], polynomial activation networks [BP21], transformers with self-attention layers [SEO⁺22b], and generative adversarial networks (GANs) [SEO⁺22a]. In these cases, the final model can be seen as the (unique) optimal solution to a convex optimization problem involving the training data.

However, the trained model is often not the mathematically optimal model, as (1) most training algorithms terminate when the model is "sufficiently good", i.e., before reaching the optimal model, and (2) the optimal model may not be representable using finite precision numbers, which means any model represented on computers is only an approximation anyway.

In practice, we can thus show that the trained model is close to the optimal model; i.e., we can incorporate a notion of proximity by proving some bounds on the difference between a given model and the optimal model implied by the training data. Thus, we propose the notion of ϵ -vicinity: a model w is within the ϵ -vicinity if $||w - w^*||_2 \le \epsilon$, where w^* is the mathematically optimal model. Our definition has a few direct implications. First, the definition is independent of how the model is trained in terms of the hardware platform or training algorithm. Proving this property also requires no changes to the training process. Second, for machine learning models where the predictions are of the form of f(x, w) for some K-Lipschitz function f and data point x, this bound implies similarity in the classification result as well, as $||f(x, w) - f(x, w^*)||_2 \le K \cdot \epsilon$. However, this definition is non-constructive in the sense that while w^* always exists for convex optimizations, it is not always computable in its precise form.

Verifying Optimum Vicinity with Real-Number Operations. To summarize, thus far, we are given the training data D, which decides the loss function \mathcal{L}_D that takes in any model and outputs a score. We know that there exists a model w^* that minimizes \mathcal{L}_D and want to verify that a model w is ϵ -vicinal to w^* . The challenge, however, arises in considering how to verify this without explicit calculation of w^* .

Our second insight is that although it is difficult (and seemingly impossible) to find the exact distance between w and w^* , one can find an upper bound instead. In particular, we do know that $\mathcal{L}_D(w^*)$ is a global minimum, meaning its gradient $\nabla \mathcal{L}_D(w^*) = 0$. As such, we can relate the distance between w^* and wwith the distance between $\nabla \mathcal{L}_D(w^*)$ and $\nabla \mathcal{L}_D(w)$ when the loss function \mathcal{L}_D is "well behaved". To be more precise, it can be shown that when the loss function is *m*-strongly convex, we have that

$$\|\boldsymbol{w} - \boldsymbol{w}^*\|_2 \leq \frac{\|\nabla \mathcal{L}_D(\boldsymbol{w})\|_2}{m},$$

where m may not have an explicit formula and, even when it does, may depend on w^* . Therefore, our focus is to, again, get a meaningful bound on m. We observe that when the loss function includes a regularization term $(\lambda/2) \cdot ||w||_2^2$, m can be lower bounded by λ . This means that we have

$$\|oldsymbol{w}-oldsymbol{w}^*\|_2 \leq rac{\|
abla \mathcal{L}_D(oldsymbol{w})\|_2}{\lambda}$$

Because of the two relaxations in optimum vicinity up to this point, it is possible that the prover has a w satisfying ϵ -vicinity but is able to prove only ϵ' -vicinity, where $\epsilon' > \epsilon$. We show in Section 5.3 that such a gap is usually small ($\epsilon' - \epsilon \le 0.0001$) when λ is in a reasonable range ($\lambda \ge 0.25$). In practice, the prover can always train a model so that the gradient is even smaller.

ZK Proof of Optimum Vicinity Without Real-Number Operations. So far, we assume that the verification operates on real numbers, which is mathematically sound but cannot be realized on modern computers or within cryptographic protocols. To enable ZKPs of optimum vicinity, we need to translate this bound to computation on bounded arithmetic. Our high-level idea is to perform a bounded approximation of the above formula while keeping track of the error. If the error always overapproximates by some value Δ_{fp} , the resulting proof would be ($\epsilon + \Delta_{fp}$)-vicinity, i.e., it would hurt tightness but not soundness. Therefore, our final task is to efficiently and accurately overapproximate the computation of the gradient of the loss function.

To this end, we describe in Section 4 two contributions. First, we implement fixed-point interval arithmetic in zero knowledge to keep track of the upper and lower bounds of every value so that we can propagate the bound as the computation proceeds. This ensures that all intermediate results are strict overestimations of their real values, thereby preserving soundness. Second, we present a precise approximation of the sigmoid function, specifically optimized for interval arithmetic and efficiently verifiable in zero knowledge. All existing sigmoid approximations lack sufficient accuracy and are incompatible with interval arithmetic because they fail to maintain both lower and upper bounds. In contrast, our approximation uses two piecewise linear functions to separately approximate the lower and upper bounds of the sigmoid. This dual-approximation strategy guarantees compatibility with interval arithmetic while also minimizing approximation error. As we show in Section 5.4, by carefully combining these two techniques, we are able to maintain soundness, achieve small overestimation errors, and remain efficient in the final proof.

Summary. Figure 1 summarizes our approach to proving ϵ -vicinity. To prove that a model w is ϵ -vicinal, we establish an upper bound ϵ_{fp} in the proof that overestimates its real distance (ϵ_{real}) to the optimal model w^* . This upper bound is the result of a series of approximation techniques, which introduce three gaps:

- 1. Strong convexity gap: $\Delta_{sc} = |\epsilon_{sc} \epsilon_{real}|$ (Section 3.2). Since w^* is unknown, we upper-bound ϵ_{real} using the strong convexity of the loss function, resulting in ϵ_{sc} .
- 2. **Regularization gap:** $\Delta_{rg} = |\epsilon_{reg} \epsilon_{sc}|$ (Section 3.3). The strong convexity parameter *m* is also infeasible to compute. Therefore, we further upper-bound ϵ_{sc} with ϵ_{reg} by substituting *m* with the regularization parameter λ .



Figure 1: Roadmap for bounding ϵ -vicinity. To prove that a model w is ϵ -vicinal, our approach establish a bound (ϵ_{fp}) on its exact distance to the optimal model w^* (ϵ_{real}), This process introduces three gaps between ϵ_{real} and ϵ_{fp} . We maintain soundness and restrain the overestimation error for each gap through a series of targeted techniques.

3. Fixed-point gap: $\Delta_{fp} = |\epsilon_{fp} - \epsilon_{reg}|$ (Section 4). To ensure soundness against fixed-point precision errors, we employ fixed-point interval arithmetic and an accurate sigmoid approximation in the final proof to obtain ϵ_{fp} .

Throughout the paper, we demonstrate how each gap is managed in a sound and tight manner. We stress that the approximation errors introduced by these three gaps only inflate the bound, resulting in a loss of tightness but *not* a loss of soundness. Specifically, since $\epsilon_{fp} \ge \epsilon_{real}$, we guarantee that the real distance of w to the optimal model w^* is within the proved bound. Our experiments (Section 5) further confirm that this tightness loss is practically small, ensuring that ϵ_{fp} remains a close and reliable approximation of ϵ_{real} .

Evaluation. In Section 5, we evaluate our protocol using five real-world datasets for both logistic regression and soft-margin SVMs. Our experimental results demonstrate that the protocol: (1) is robust against rejection sampling attacks, with ϵ -vicinal models preserving nearly all confident predictions of the optimal model; (2) maintains sound approximation with a low overestimation error at each step (approximately 10^{-4} and 10^{-3} for $\Delta_{sc} + \Delta_{rg}$ and Δ_{fp} , respectively); and (3) significantly reduces the Boolean circuit size, up to $246 \times$ smaller than the current zkPoT paradigm, which verifies the entire training process in zero knowledge.

2 Preliminaries

In this section, we present notation and definitions of zero-knowledge proof, machine learning, and convex functions. Additional mathematical definitions and properties are provided in Appendix A.

2.1 Notation

We use bold lower-case letters like x to denote column vectors. For clarity, we sometimes denote the inner product of vectors using $\langle x, w \rangle = x^{\top} w$. We denote the ℓ^2 norm of vector w by $||w||_2 = \sqrt{w^{\top} w}$. We define signed fixed-point numbers as real numbers represented using a fixed scaling factor. Specifically, we represent a signed fixed-point number as a signed (m + n)-bit integer with m integer bits and n fractional bits. The real value r is given by $r = x \cdot 2^{-n}$.

2.2 Zero-Knowledge Proofs

Zero-knowledge proofs (ZKPs) for circuit satisfiability allow a prover \mathcal{P} holding a witness w to prove to a verifier \mathcal{V} that $\mathcal{C}(w) = 1$ for some public circuit \mathcal{C} without revealing any information about w. The goal of this paper is not to design new ZKPs but to use existing protocols to construct an efficient and meaning-ful zero-knowledge proof of training. Therefore, we abstract the ZKP functionality as shown in Figure 2. Essentially, we require that the prover can commit to its input first and then prove a statement about the committed input, along with auxiliary inputs. Most ZKP protocols can instantiate this functionality efficiently,

Functionality \mathcal{F}_{ZK}

Commit: Upon receiving (commit, w) from prover \mathcal{P} and (commit) from verifier \mathcal{V} , the functionality stores w. **Prove:** Upon receiving (prove, \mathcal{C}) from \mathcal{P} and (prove, \mathcal{C}) from \mathcal{V} where \mathcal{C} is a circuit, retrive w from memory and check if $\mathcal{C}(w)$ equals 1. The functionality aborts if the circuits from two parties are different or if $\mathcal{C}(w) \neq 1$; otherwise, send true to both \mathcal{P} and \mathcal{V} .

Figure 2: Functionality for ZK proofs of circuit satisfiability.

including (but not limited to) zkSNARKs [Gro16, CHM⁺20, GWC19, Lab17] and proofs based on vector oblivious linear evaluation (VOLE) [WYY⁺22, BMRS21, DILO22, BBMHS22, BBMH⁺21, YSWW21].

2.3 Machine Learning Preliminaries

We consider a generalized framework for training machine learning models. While this paper primarily focuses on logistic regression and SVMs as examples, our results can be extended to other types of models where training can be formulated as an optimization problem that minimizes a loss function.

Specifically, a machine learning model is parameterized by a vector $\boldsymbol{w} \in \mathbb{R}^d$, which we interchangeably refer to as the model or the model parameters. Let $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ denote the training dataset, where $\boldsymbol{x}_i \in \mathbb{R}^d$ represents input features and $y_i \in \mathbb{R}$ the corresponding labels. Let \mathcal{R} denote the space of randomness, and \mathcal{L}_D denotes a loss function. A training algorithm is defined as a mapping $\text{Train}_{\mathcal{L}_D} : D \times \mathcal{R} \to \mathbb{R}^d$, where $\text{Train}_{\mathcal{L}_D}(D, r)$ takes as input a training dataset D and randomness $r \in \mathcal{R}$, and outputs a trained model \boldsymbol{w} . The training algorithm aims to find a model \boldsymbol{w} that minimizes the loss function $\mathcal{L}_D : \mathbb{R}^d \to \mathbb{R}$. Therefore, the general training objective can be formulated as:

$$\min_{oldsymbol{w}\in\mathbb{R}^d} \quad \mathcal{L}_D(oldsymbol{w}).$$

When the dataset D is fixed or implied from the context, we may denote the loss function simply as $\mathcal{L}(w)$. We now define a few models that appear later in the paper.

Logistic regression. Logistic regression is a common machine learning model for binary classification tasks with inputs $x_i \in \mathbb{R}^d$ and outputs $y_i \in \{0, 1\}$. We consider a model that predicts the label using the sigmoid function $\sigma(z) = 1/(1 + e^{-z})$. The model makes predictions by applying the sigmoid function to $w^{\top}x$ and thresholding the probability at 0.5: if $\sigma(w^{\top}x) > 0.5$, the label is predicted as 1, and 0 otherwise. Given a training dataset of i.i.d. observations $D_n := \{(x_i, y_i)\}_{1 \le i \le n}$, logistic regression learns the parameters w by minimizing the logistic loss function:

$$\mathcal{L}_{\text{logit}}(\boldsymbol{w}) = -\sum_{i=1}^{n} \left[y_i \log \sigma \left(\boldsymbol{w}^{\top} \boldsymbol{x}_i \right) + (1 - y_i) \log \sigma \left(- \boldsymbol{w}^{\top} \boldsymbol{x}_i \right) \right]$$

The convexity of the logistic loss ensures that optimization algorithms, such as gradient descent, can efficiently find the global minimum.

In practice, L_2 -regularization is often applied to logistic regression to prevent overfitting. It adds a penalty term proportional to the squared norm of w:

$$\mathcal{L}_{ ext{logit}}^{\lambda}(\boldsymbol{w}) = \mathcal{L}_{ ext{logit}}(\boldsymbol{w}) + \frac{\lambda}{2} \|\boldsymbol{w}\|_{2}^{2},$$

where $\lambda > 0$ is the regularization parameter.

Soft-margin support vector machines (SVMs). For a binary classification tasks with inputs $x_i \in \mathbb{R}^d$ and outputs $y_i \in \{-1, 1\}$, SVMs predict labels using a linear decision function: $f(x_i) = w^{\top} x_i + b$, where $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are the model parameters. SVMs aim to maximize the margin between the two classes while minimizing classification errors. Soft-margin SVMs, which do not assume linear separability of the data, achieve this by minimizing a hinge loss function with an L_2 regularization term:

$$\mathcal{L}_{\text{SVM}}^{\lambda}(\boldsymbol{w}, b) = \sum_{i=1}^{n} \max(0, 1 - y_i(\boldsymbol{w}^{\top} \boldsymbol{x}_i + b)) + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2,$$

where $\lambda > 0$ is the regularization parameter. For simplicity, we treat b as a constant so the model can be represented by w alone.

2.4 Convex and Strongly Convex Functions

Our methods require the loss function of the training algorithm to be strongly convex; thus, we provide the relevant definitions here. We begin by recalling the notions of convex and strongly convex functions for \mathbb{R}^d , which form the foundation for many optimization-based machine learning methods.

Definition 1 (Convex Functions [BV04]). A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex on \mathbb{R}^d if for all $x, y \in \mathbb{R}^d$ and for all $0 \le \theta \le 1$, the following inequality holds:

$$f(\theta x + (1 - \theta)y) \le \theta f(x) + (1 - \theta)f(y).$$

A convex function ensures any local minimum is also a global minimum. However, merely knowing a function is convex does not always guarantee fast convergence. For more efficient optimization, we often seek stronger conditions on the function's curvature, which brings us to the definition of strong convexity.

Definition 2 (Strongly Convex Functions [BV04]). A continuously differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is *m*-strongly convex on \mathbb{R}^d with parameter m > 0 if for any $x, y \in \mathbb{R}^d$, the following inequality holds:

$$f(y) \ge f(x) + \nabla f(x)^{\top} (y - x) + \frac{m}{2} \|y - x\|_2^2.$$

Alternatively, if f is twice continuously differentiable, f is m-strongly convex if and only if its Hessian satisfies:

$$\nabla^2 f(x) \succeq m I_d \quad \forall x \in \mathbb{R}^d,$$

where I_d is the d-dimensional identity matrix.

Strong convexity gives us many appealing properties for optimization. Intuitively, it guarantees that the function grows as fast as a quadratic function, which leads to faster convergence rates in gradient-based methods. Moreover, it ensures that if gradients change minimally when moving from point x to point y, then the distance between x and y remains small. This insight also justifies common stopping criteria in machine learning, where a training algorithm terminates if the gradient becomes sufficiently small.

Formally, we have the following lemma capturing this idea. It is a direct implication of strong convexity.

Lemma 1 (Implication of Strong Convexity [Zho18]). Let $f : \mathbb{R}^d \to \mathbb{R}$ be a continuously differentiable, *m*-strongly convex function. Then, for any $x, y \in \mathbb{R}^d$,

$$||x - y||_2 \le \frac{1}{m} ||\nabla f(x) - \nabla f(y)||_2$$

3 Defining and Verifying Optimum Vicinity

In this section, we propose *optimum vicinity*, a new definition for proofs of training. Along with the definition, we also present a practical method for checking the optimum vicinity.

We first formally define ϵ -vicinal models and show how their performance is guaranteed in Section 3.1. The definition, however, is non-constructive as it depends on the mathematically optimal model, which is generally not computable. To address this limitation, we derive an upper bound on the distance between the ϵ -vicinal model and the optimal model in Section 3.2, assuming *m*-strong convexity of the loss function. This bound directly relies on the strong convexity parameter *m*, which is often unknown. In Section 3.3, we further prove an upper bound for *m*, allowing for verification of a model's ϵ -vicinity without knowing the exact optimal solution or the value of the strong convexity parameter.

3.1 Defining Optimum Vicinity

We define optimal vicinity based on the proximity of a given model to the optimal model. Intuitively, an optimal model minimizes a specified loss function, and a model is considered ϵ -vicinal if its distance from an optimal model is at most ϵ .

Definition 3 (ϵ -vicinity). A model w^* is optimal w.r.t. a loss function \mathcal{L} if $w^* \in \arg \min_{w} \mathcal{L}(w)$. A model $w \in \mathbb{R}^d$ is ϵ -vicinal w.r.t. an optimal model w^* if $||w - w^*||_2 \leq \epsilon$.

We denote the exact distance $||w - w^*||_2$ by ϵ_{real} . By our definition, any model w that is ϵ_{real} -vicinal is also ϵ' -vicinal for any $\epsilon' \ge \epsilon_{real}$. This property will be leveraged to derive a series of upper bounds on ϵ -vicinity later in the paper. Note an optimal model is always defined w.r.t. a fixed loss function \mathcal{L} , and ϵ -vicinity is measured relative to this optimal model w^* . For clarity, we may omit explicit references to \mathcal{L} or w^* when the context is clear.

Implications of ϵ -vicinity. ϵ -vicinity provides a quantifiable measure of how closely a given model approximates the optimal model. Theorem 1 further demonstrates that an ϵ -vicinal model with a sufficiently small ϵ is guaranteed to achieve near-optimal performance. Specifically, an ϵ -vicinal model exhibits only minimal prediction error, provided the prediction function is Lipschitz continuous w.r.t. the model's parameter w.

Theorem 1 (Prediction error bound for ϵ -vicinal models). Suppose the prediction of an ϵ -vicinal model w and an optimal model w^* on a data point x can be written as f(w, x), where $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is *K*-Lipschitz continuous w.r.t. w. Then, it follows that

$$|f(\boldsymbol{w}, \boldsymbol{x}) - f(\boldsymbol{w}^*, \boldsymbol{x})| \leq K \cdot \epsilon.$$

Proof. By the parameter-wise K-Lipschitz continuity (Definition 5) of f we have

$$|f(\boldsymbol{w},\boldsymbol{x}) - f(\boldsymbol{w}^*,\boldsymbol{x})| \le K \cdot \|\boldsymbol{w} - \boldsymbol{w}^*\|_2.$$

Because w is ϵ -vicinal, we have $||w - w^*||_2 \le \epsilon$. Combining these results, we have

$$|f(\boldsymbol{w}, \boldsymbol{x}) - f(\boldsymbol{w}^*, \boldsymbol{x})| \le K \cdot \|\boldsymbol{w} - \boldsymbol{w}^*\|_2 \le K \cdot \epsilon,$$

which completes the proof.

According to Theorem 1, verifying that a model is ϵ -vicinal with small ϵ is sufficient to guarantee that its predictions closely approximate those of the optimal model.

For most common linear models like logistic regression or SVMs, the model's predictions can be expressed as a function of $\langle w, x \rangle$. In this case, we can derive the following corollary:

Corollary 1 (Prediction error bound for linear predictor function.). Suppose the prediction can be represented as $f(w, x) = g(\langle w, x \rangle)$, where g is K-Lipschitz continuous in its scalar argument. Then, it follows that

$$|f(\boldsymbol{w},\boldsymbol{x}) - f(\boldsymbol{w}^*,\boldsymbol{x})| \le K \cdot \|\boldsymbol{x}\|_2 \cdot \epsilon.$$

Proof. We first prove $f(w, x) = g(\langle w, x \rangle)$ is $(K \cdot ||x||_2)$ -Lipschitz w.r.t. w. Then, the corollary follows from Theorem 1. To see this, notice because g is K-Lipschitz continuous, we have

$$g(\langle \boldsymbol{w}, \boldsymbol{x} \rangle) - g(\langle \boldsymbol{w}^*, \boldsymbol{x} \rangle) \leq K \cdot |\langle \boldsymbol{w}, \boldsymbol{x} \rangle - \langle \boldsymbol{w}^*, \boldsymbol{x} \rangle| = K \cdot \langle \boldsymbol{w} - \boldsymbol{w}^*, \boldsymbol{x} \rangle.$$

Applying the Cauchy-Schwarz inequality on the right-hand side, we have

$$K \cdot \langle \boldsymbol{w} - \boldsymbol{w}^*, \boldsymbol{x} \rangle \leq K \cdot \|\boldsymbol{x}\|_2 \cdot \|\boldsymbol{w} - \boldsymbol{w}^*\|_2.$$

Combining all, we have

$$g(\langle \boldsymbol{w}, \boldsymbol{x} \rangle) - g(\langle \boldsymbol{w}^*, \boldsymbol{x} \rangle) \leq K \cdot \|\boldsymbol{x}\|_2 \cdot \|\boldsymbol{w} - \boldsymbol{w}^*\|_2.$$

We conclude f is $(K \cdot ||\boldsymbol{x}||_2)$ -Lipschitz with respect to \boldsymbol{w} .

This corollary indicates that the prediction error depends on the norm $||\mathbf{x}||_2$ of the data point. For example, the prediction function from a logistic regression model is $f(\mathbf{w}, \mathbf{x}) = \sigma(\langle \mathbf{w}, \mathbf{x} \rangle)$. Since σ is 0.25-Lipschitz continuous (its derivative is upper bounded by 0.25), Corollary 1 gives us

$$|f(\boldsymbol{w}, \boldsymbol{x}) - f(\boldsymbol{w}^*, \boldsymbol{x})| \le 0.25 \cdot \|\boldsymbol{x}\|_2 \cdot \epsilon.$$

In practice, datasets are typically normalized to ensure that $||x||_2$ is small. This normalization also helps keep the error bound tight, as we demonstrate empirically in Section 5.2 (Figure 7).

3.2 Bounding ϵ -Vicinity via Strong Convexity

Up to this point, we defined ϵ -vicinal models and showed that this concept provides a measure of how closely a given model approximates the optimal model. In practice, however, the optimal model is often unavailable or cannot be exactly represented due to finite precision. This makes it infeasible to compute a model's exact distance, ϵ_{real} , from the optimal model.

To overcome this issue, we introduce a method to bound ϵ_{real} for any given model under the assumption of a strongly convex loss function. This approach allows us to directly evaluate how close the model is to the optimal model without requiring knowledge of the optimal model.

Theorem 2 (Upper bound for ϵ -vicinity). Let w^* be an (unique) optimal model w.r.t. a continuously differentiable, *m*-strongly convex loss function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$. Then, any model $w \in \mathbb{R}^d$ is ϵ -vicinal w.r.t. w^* , where

$$\epsilon = \frac{1}{m} \|\nabla \mathcal{L}(\boldsymbol{w})\|_2.$$

Proof. Since \mathcal{L} is continuously differentiable and *m*-strongly convex, the optimal model w^* satisfies $\nabla \mathcal{L}(w^*) = 0$. Applying Lemma 1 with x = w and $y = w^*$, we obtain

$$\|\boldsymbol{w} - \boldsymbol{w}^*\|_2 \leq \frac{1}{m} \|\nabla \mathcal{L}(\boldsymbol{w}) - \nabla \mathcal{L}(\boldsymbol{w}^*)\|_2 = \frac{1}{m} \|\nabla \mathcal{L}(\boldsymbol{w})\|_2$$

Therefore, the model \boldsymbol{w} is ϵ -vicinal with $\epsilon = (1/m) \cdot \|\nabla \mathcal{L}(\boldsymbol{w})\|_2$.

Theorem 2 states that if the loss function is *m*-strongly convex, we can derive an upper bound on model ϵ -vicinity by using the gradient norm at \boldsymbol{w} and m. We denote this upper bound of by ϵ_{sc} , where $\epsilon_{real} \leq \epsilon_{sc} = (1/m) \cdot \|\nabla \mathcal{L}(\boldsymbol{w})\|_2$. The theorem provides a method for checking ϵ -vicinity that has several desirable properties:

- Checking ϵ -vicinity does not require access to the optimal model. It can be achieved by checking whether $\epsilon_{sc} \leq \epsilon$ holds, without knowing the optimal model.
- The ϵ -vicinity of a model can be verified independently of the training algorithm used to obtain w, enabling a clear separation between the proof of training and the training process itself.
- Verification of ϵ -vicinity is deterministic and thus requires no randomness.

Based on these properties, we proceed to formulate a meaningful definition of proof of training derived from model ϵ -vicinity.

3.3 **Proof of Training via Optimum Vicinity.**

Theorem 2 gives us a way to verify model ϵ -vicinity even when the optimal model is unknown. Assume a prover \mathcal{P} trained a model $w \leftarrow \text{Train}_{\mathcal{L}}(D, r)$ on dataset D using a m-strongly convex loss function \mathcal{L} . A verifier \mathcal{V} can check whether w is ϵ -vicinal by checking the condition:

$$\epsilon \geq \epsilon_{\mathsf{sc}} = \frac{1}{m} \|\nabla \mathcal{L}(\boldsymbol{w})\|_2.$$

In our context, since m is dependent on the loss function \mathcal{L} , we need an explicit way to compute m. However, no explicit formula to compute m is generally known [BV04]. For cases when there is a formula, it is dependent on the optimal model w^* .

For example, consider the one-dimensional logistic loss function

$$\mathcal{L}(w) = -\sum_{i=1}^{n} \Big[y_i \log \sigma(wx_i) + (1 - y_i) \log \big(1 - \sigma(wx_i) \big) \Big].$$

Its second derivative is

$$\mathcal{L}''(w) = \sum_{i=1}^{n} x_i^2 \,\sigma\big(wx_i\big)\big(1 - \sigma\big(wx_i\big)\big).$$

Since $\mathcal{L}''(w)$ is always positive (assuming not all $x_i = 0$), the loss function is locally strongly convex within any closed interval [Bac14]. To determine m in ϵ_{sc} for a model w', one must find the minimal value of $\mathcal{L}''(w)$ over the interval $w \in [w', w^*]$. Notice that $\mathcal{L}''(w)$ is bell-shaped, which ensures that its minimal value occurs at one of the endpoints of the interval. Consequently, we can express m as:

$$m = \min\{\mathcal{L}''(w'), \mathcal{L}''(w^*)\}.$$

We now circle back to the original challenge of not knowing w^* . Thus, additional techniques are required to estimate or bound m.

Bounding *m* with regularization. Our key observation is that we can obtain a lower bound of *m* by incorporating a regularization term into the loss function. A regularization term, widely used in machine learning, is added to the loss function to prevent overfitting. We notice that adding an L_2 regularization term $(\lambda/2) \cdot ||\boldsymbol{w}||_2^2$ will make any convex loss function λ -strongly convex. Therefore, $\lambda \leq m$ serves as a lower

bound for the strong convexity parameter m. With this lower bound in place, the verifier can directly check model ϵ -vicinity by verifying whether $\epsilon \geq \frac{1}{\lambda} \|\nabla \mathcal{L}(\boldsymbol{w})\|_2$, without knowing m. We denote this new bound by ϵ_{reg} . This verification process remains sound because the following inequalities hold:

$$\epsilon \geq \epsilon_{\mathsf{reg}} = \frac{1}{\lambda} \| \nabla \mathcal{L}(\boldsymbol{w}) \|_2 \geq \frac{1}{m} \| \nabla \mathcal{L}(\boldsymbol{w}) \|_2.$$

Impact of λ **on bound tightness.** While our method of using λ to bound *m* is sound, the value of ϵ_{reg} is inversely proportional to the value of λ . In other words, the choice of λ directly affects the tightness of our bound. If λ is chosen to be too large solely to achieve a tight bound, it can degrade model performance by over-penalizing it. Therefore, it is crucial that our bound remains tight, i.e., $\epsilon_{reg} - \epsilon_{real}$ is small, when λ is picked from a wide range of values. Fortunately, as we show in Section 5.3, this gap remains small for a wide range of commonly used λ . This ensures that a prover can train their model with various λ values and still prove its ϵ -vicinity with a tight bound.

Optimum vicinity for real-world models. We stress that the above technique is applicable to a wide range of real-world models as long as the loss function is convex. By adding an L_2 regularization term to the convex loss, it becomes strongly convex, and λ is a lower bound of the strong convexity parameter m.

In this paper, we pick logistic regression and soft-margin SVMs as two motivating real-world examples. Despite their relative simplicity, logistic regression and SVMs are widely used in numerous applications [NWH⁺14, YYH03] and, for this reason, have frequently served as canonical examples in prior zkPoT work [GGJ⁺23, STC⁺24]. We present the theorems deriving their ϵ -vicinity below.

Theorem 3 (ϵ -vicinity bound for logistic regression). Let w^* be the optimal model w.r.t. the L_2 -regularized logistic loss function $\mathcal{L}^{\lambda}_{logit}$ (Section 2.3). Any logistic regression model $w \in \mathbb{R}^d$ is ϵ -vicinal w.r.t. w^* , where

$$\epsilon = rac{1}{\lambda} \|
abla \mathcal{L}_{logit}^{\lambda}(\boldsymbol{w}) \|_2.$$

Theorem 4 (ϵ -vicinity bound for SVMs). Let w^* be the optimal model w.r.t. the L_2 -regularized hinge loss function $\mathcal{L}_{SVM}^{\lambda}$ (Section 2.3). Any SVM $w \in \mathbb{R}^d$ is ϵ -vicinal w.r.t. w^* , where

$$\epsilon = rac{1}{\lambda} \|
abla \mathcal{L}^{\lambda}_{SVM}(oldsymbol{w}) \|_2.$$

Sketch of Proof. Notice that logistic loss and hinge loss are both convex and therefore, with the regularization term $(\lambda/2) \cdot \|\boldsymbol{w}\|_2^2$, both $\mathcal{L}_{logit}^{\lambda}$ and $\mathcal{L}_{SVM}^{\lambda}$ become λ -strongly convex. The rest of the proof follows from Theorem 2. We present the full proof for both theorems in Appendix B.

Generality of our approach. Our analysis assumes Lipschitz continuity for the prediction function and strong convexity for the loss function. We remark that these properties are commonly found in practical machine learning models, allowing our approach to be readily applied to a wide range of real-world training tasks. For example, SVMs and many generalized linear models have Lipschitz-continuous prediction functions, and virtually all loss functions in machine learning are strongly convex through standard L_2 regularization.

4 ZK Proofs of Optimum Vicinity with Finite-Precision Arithmetic

This section introduces our framework for proving model ϵ -vicinity in zero knowledge, completing our zkPoT definition. In Section 4.1, we define the ideal functionality for ϵ -vicinity verification and present a

Functionality \mathcal{F}_{VIC-ZK}

Public input: Training algorithm $\text{Train}_{\mathcal{L}}$.

Commit: Upon receiving (commit, D) from prover \mathcal{P} and commit from verifier \mathcal{V} , store D.

Prove vicinity: Upon receiving (prove, w, ϵ) from prover \mathcal{P} and prove from verifier \mathcal{V} , where w is the trained model, the functionality does the following:

- 1. Fetch the training data D, if no data is stored, abort.
- 2. Compute the optimal model w^* from D.
- 3. Check if $\|\boldsymbol{w}^* \boldsymbol{w}\|_2 \leq \epsilon$. If not, the functionality aborts; otherwise, send (checked, ϵ) to both \mathcal{P} and \mathcal{V} .

Figure 3: Ideal functionality for proving model ϵ -vicinity in zero-knowledge

Protocol Π_{VIC-ZK}

Commit: \mathcal{P} sets $w' = (w, \mathcal{D})$ and sends (commit, w') to \mathcal{F}_{ZK} . \mathcal{V} sends commit to \mathcal{F}_{ZK} . **Prove vicinity:**

- 1. Both parties agree on ϵ , a strongly convex loss function $\mathcal{L}(\cdot)$, and its strong convexity parameter m.
- 2. Let $\mathcal{C}_{\mathcal{L},\epsilon,m}$ be a boolean circuit where $\mathcal{C}_{\mathcal{L},\epsilon,m}(\boldsymbol{w},D) = 1$ if and only if $\epsilon \geq (1/m) \cdot \|\nabla \mathcal{L}_D(\boldsymbol{w})\|_2$. \mathcal{P} sends (prove, $\mathcal{C}_{\mathcal{L},\epsilon,m}$) to $\mathcal{F}_{\mathsf{ZK}}$ and \mathcal{V} sends (verify, $\mathcal{C}_{\mathcal{L},\epsilon,m}$) to $\mathcal{F}_{\mathsf{ZK}}$.
- 3. \mathcal{V} and \mathcal{P} receive output $\in \{\text{true}, \bot\}$ from $\mathcal{F}_{\mathsf{ZK}}$ and both output (checked, ϵ) if and only if output = true; otherwise both party output \bot and halt.

Figure 4: The protocol Π_{VIC-ZK} realizing \mathcal{F}_{VIC-ZK} in the \mathcal{F}_{ZK} -hybrid model.

protocol to securely realize it. In Section 4.2, we discuss the challenge of ensuring soundness under finite numerical precision. To address this challenge, we introduce techniques based on interval arithmetic (Section 4.3) and a novel, efficient sigmoid approximation (Section 4.4). These methods collectively maintain the soundness and improve the efficiency of our zkPoT protocol.

4.1 Proving Model *ε*-Vicinity in ZK

To prove model ϵ -vicinity in ZK, we essentially just need to represent the verification process in a circuit and prove using any ZK protocol that can instantiate \mathcal{F}_{ZK} . Figure 3 shows our ideal functionality for proving ϵ -vicinity in zero-knowledge. We note that the training algorithm, loss function \mathcal{L} , and its strong convexity parameter m are public, and the trained model w and training dataset D are private inputs for the prover. However, this assumes that the ZK backend can prove real-number operations, which is impossible. Below, we first provide the protocol steps and discuss how to avoid real-number operations in ZK while still preserving soundness.

Our protocol realizing this functionality in the $\mathcal{F}_{\mathsf{ZK}}$ -hybrid model is presented in Figure 4. The prover first commits to the model w and dataset D. Then, both parties agree on a circuit $\mathcal{C}_{\mathcal{L},\epsilon,m}$ where $\mathcal{C}_{\mathcal{L},\epsilon,m}(w; D) =$ 1 iff. $\epsilon \ge (1/m) \cdot \|\nabla \mathcal{L}_D(w)\|_2$. Finally, both parties send $\mathcal{C}_{\mathcal{L},\epsilon,m}$ to $\mathcal{F}_{\mathsf{ZK}}$ and receive (checked, ϵ) if the model passes the verification. We stress in practice that the m is replaced with the regularization parameter λ as the exact m is unknown (see Section 3.3).

The protocol verifies in ZK whether the claimed ϵ satisfies the upper bound of ϵ -vicinity derived in Theorem 2. Since ϵ_{sc} represents an upper bound on the true model ϵ -vicinity, any valid ϵ satisfying $\epsilon \geq \epsilon_{sc}$ guarantees that the committed model w is indeed ϵ -vicinal. The security of Π_{VIC-ZK} directly reduces to

the security of \mathcal{F}_{ZK} , as it uses \mathcal{F}_{ZK} to prove circuit satisfiability in a straightforward manner. We state the following theorem about our protocol security and provide the proof in Appendix **B**.

Theorem 5 (Protocol security). The protocol Π_{VIC-ZK} in Figure 4 securely realizes the ideal functionality \mathcal{F}_{VIC-ZK} (Figure 3) in the \mathcal{F}_{ZK} -hybrid model.

4.2 Maintaining Soundness under Finite Numerical Precision

While the protocol outlined in the previous section establishes a secure zkPoT, its soundness relies on the assumption that all operations are performed over real numbers with infinite numerical precision. However, all cryptographic implementations use finite numerical precision formats (e.g., fixed-point numbers), which could introduce numerical errors and compromise soundness. For example, the upper bound of an ϵ -vicinal model is given by $\epsilon \leq \epsilon_{\text{reg}}$, where ϵ_{reg} is a real number. Suppose its fixed-point representation is $\epsilon_{\text{fp}} = \epsilon_{\text{reg}} + \Delta_{\text{fp}}$ where Δ_{fp} is the numerical error. If $\Delta_{\text{fp}} < 0$, then $\epsilon_{\text{fp}} < \epsilon_{\text{reg}}$ and the inequality $\epsilon \leq \epsilon_{\text{fp}}$ might no longer hold.

To address this issue, we compute an overapproximated bound $\epsilon_{fp} \ge \epsilon_{reg}$ in fixed-point. The computation of this bound for logistic regression requires performing addition, subtraction, multiplication, and the evaluation of the sigmoid function, each of which is handled independently. Our solution consists of two main components:

- 1. Fixed-point interval arithmetic. We first ensure the result of all basic fixed-point operations (addition, subtraction, and multiplication) is an overapproximation of the real value. This can be achieved using interval arithmetic [HJVE01], where every real value a is represented as a fixed-point interval $[a_{lo}, a_{hi}]$ with the invariant $a_{lo} \le a \le a_{hi}$. Interval arithmetic is defined to ensure all operations preserve this invariant. We use interval arithmetic in ZK to perform all intermediate computations, ensuring that the final interval provides a sound overapproximation of the bound for ϵ -vicinity.
- 2. Accurate sigmoid approximation. For sigmoid, although we could theoretically directly apply interval arithmetic as well to achieve soundness, it would still be too expensive to prove in ZK. Instead, we design an efficient and accurate sigmoid approximation function tailored for interval arithmetic. Our technique maintains the invariant of interval arithmetic and is efficient to compute in ZK.

To ensure our bound computed in fixed-point arithmetic is both sound and tight, our solution must meet these two criteria:

• Soundness: The fixed-point bound must always be greater than or equal to the true bound.

• **Tightness:** The fixed-point bound should have minimal approximation error $\Delta_{fp} = \epsilon_{fp} - \epsilon_{reg}$.

We now present our techniques in detail and show how they collectively ensure soundness and tightness for fixed-point numbers.

4.3 Fixed-Point Interval Arithmetic

Interval arithmetic for real intervals. Interval arithmetic is a mathematical technique designed to mitigate the accumulated errors inherent in finite precision computations. Instead of representing an exact real value with a single limited-precision number, interval arithmetic associates the value with a closed real interval. This interval contains all possible real numbers that the exact value could represent, and the interval endpoints can be viewed as the lower and upper bound of the real value. For example, a real number $x \in \mathbb{R}$ is associated with an interval $[x_{lo}, x_{hi}]$, where $x_{lo}, x_{hi} \in \mathbb{R}$ and $x_{lo} \leq x \leq x_{hi}$. This interval provides a robust way to account for errors in finite precision computations. Arithmetic operations on intervals are defined to ensure correctness. For example, given $x \in [x_{lo}, x_{hi}]$ and $y \in [y_{lo}, y_{hi}]$, the operation $[x_{lo}, x_{hi}] \circ [y_{lo}, y_{hi}] = [z_{lo}, z_{hi}]$ must ensure that $x \circ y \in [z_{lo}, z_{hi}]$.

For $a, b, c, d \in \mathbb{R}$, interval addition, subtraction, and multiplication can be defined as follows [HJVE01]: • Interval addition. [a, b] + [c, d] = [a + c, b + d].

• Interval subtraction. [a, b] - [c, d] = [a - d, b - c].

• Interval multiplication. $[a, b] * [c, d] = [\min(S), \max(S)]$, where $S = \{a * c, a * d, b * c, b * d\}$.

Interval arithmetic for fixed-point intervals. For our purposes, we employ interval arithmetic on fixedpoint to ensure that the final fixed-point bound is guaranteed to be greater than or equal to the corresponding real value, i.e., $\epsilon_{fp} \ge \epsilon_{real}$. Since fixed-point numbers are a subset of reals, extending interval arithmetic to fixed-point intervals is straightforward. However, one caveat is that fixed-point arithmetic can be imprecise due to rounding, which introduces precision errors even when using interval arithmetic. This can be solved using different rounding modes for the lower and upper bounds. Specifically, let $\mathbb{FP}(m, n) \subseteq \mathbb{R}$ denote the set of all reals representable by signed fixed-point numbers with m integer bits and n fractional bits. Let $*_{lo}, *_{hi}$ denote fixed-point multiplication over $\mathbb{FP}(m, n)$ with rounding towards $-\infty$ (round down) and $+\infty$ (round up), respectively. Let $a, b, c, d \in \mathbb{FP}(m, n)$. We define addition, subtraction, and multiplication of fixed-point intervals as follows:

- Interval addition. [a, b] + [c, d] = [a + c, b + d].
- Interval substraction. [a, b] [c, d] = [a d, b c].
- Interval multiplication. $[a, b] * [c, d] = [\min(S), \max(S)]$, where $S = \{a *_{lo} c, a *_{lo} d, b *_{lo} c, b *_{lo} d, a *_{hi} c, a *_{hi} d, b *_{hi} c, b *_{hi} d\}$.

The correctness of addition and subtraction stems from the fact that fixed-point operations in these cases are exact, provided no overflow occurs. For multiplication, the correctness follows from rounding conservatively for both the lower and upper bound. We refer to [HJVE01] for a comprehensive discussion of finite-precision interval arithmetic and detailed proofs.

Using fixed-point interval arithmetic, we can guarantee that the final fixed-point interval provides the lower and upper bound of the real value. Then, we simply use the upper bound endpoint value as our final overapproximated result.

4.4 Sigmoid Approximation for Interval Arithmetic

Theoretically, interval arithmetic can be applied to sigmoid evaluation as well because the sigmoid function is monotonic. Consider a real value $z \in [z_{lo}, z_{hi}]$. Then, since sigmoid is monotonically increasing, we have $\sigma(z) \in [\sigma(z_{lo}), \sigma(z_{hi})]$. However, directly evaluating sigmoid is still too computationally expensive in ZK. A natural and widely adopted solution is to use sigmoid approximation.

Issues with existing sigmoid approximation schemes. Previous works have explored various sigmoid approximations in cryptographic contexts [GGJ⁺23, MR18, MZ17, APKP24]. Unfortunately, all existing approaches consider a single approximation function, which fails in interval arithmetic. To see this, consider a sigmoid approximation function σ' that has approximation error δ at a point *z*:

$$\sigma'(z) = \sigma(z) + \delta.$$

Now consider the input fixed-point interval $z \in [z, z]$. Here, $z = z_{lo} = z_{hi}$, which means z is represented exactly as a fixed-point number. The output of the approximated sigmoid function over this interval is:

$$\sigma'([z,z]) = [\sigma'(z), \sigma'(z)] = [\sigma(z) + \delta, \sigma(z) + \delta].$$

Here, the true sigmoid value $\sigma(z)$ does not lie within the interval $[\sigma(z) + \delta, \sigma(z) + \delta]$ unless $\delta = 0$. This discrepancy violates the correctness requirement for interval arithmetic and makes existing sigmoid approximation schemes unsuitable for our purposes.

Our solution: efficient sigmoid approximation for interval arithmetic. The primary issue of prior sigmoid approximations is that they all use a single approximation function, so they cannot keep track of both the lower and upper bounds of the true sigmoid values at the same time. To address this issue, we propose *a pair of* sigmoid approximation functions that separately handle the lower and upper bound of the input interval. Specifically, we define two distinct functions σ_{lo} and σ_{hi} , with the following properties:

$$\sigma_{lo}(z) \le \sigma(z)$$
 and $\sigma_{hi}(z) \ge \sigma(z), \quad \forall z \in \mathbb{R}.$

In essence, σ_{lo} and σ_{hi} serve as global lower and upper bounds for the true sigmoid function, respectively. By constructing σ_{lo} and σ_{hi} to consistantly underapproximate and overapproximate the true sigmoid function σ , we ensure that for any input interval $z \in [z_{lo}, z_{hi}]$, the true sigmoid value lies within the interval:

$$\sigma(z) \in [\sigma_{lo}(z_{lo}), \sigma_{hi}(z_{hi})].$$

We provide a formal proof of correctness in Appendix C.1.

We now present the constructions of σ_{lo} and σ_{hi} . Both functions are defined as piecewise linear approximations of the sigmoid function, taking the general form:

$$f(x) = \begin{cases} a_1 x + b_1, & d_{min} \le x \le d_1, \\ a_2 x + b_2, & d_1 < x \le d_2, \\ \dots \\ a_n x + b_n, & d_{n-1} < x \le d_{max} \end{cases}$$

where d_{min} and d_{max} are the minimal and maximal values of the fixed-point representation.

To make sure σ_{lo} and σ_{hi} provide the lower and upper bound of the true sigmoid function, our construction is based on the following principles:

- When z > 0, the sigmoid is *concave*. A tangent line provides an upper bound, while a line segment connecting two points provides a lower bound.
- When z < 0, the sigmoid is *convex*. A tangent line provides a lower bound, while a line segment connecting two points provides an upper bound.

In addition, we address the behavior of the sigmoid function as it approaches its asymptotic values of 0 and 1:

- When σ(z) approaches 0, its asymptote at 0 provides a valid lower bound. Since sigmoid is monotonic, we can set σ_{hi}(z) = σ(d⁻) + δ for all z < d⁻, where δ > 0 is a small offset and d⁻ is a negative threshold value.
- When $\sigma(z)$ approaches 1, its asymptote at 1 provides a valid upper bound. For a lower bound, similarly, we set $\sigma_{lo}(z) = \sigma(d^+) \delta$ for all $z > d^+$, where $\delta > 0$ is a small offset and d^+ is a positive threshold value.

Based on these principles, we divide the sigmoid domain into four regions: $[d_{min}, d^-)$, $[d^-, 0)$, $[0, d^+]$, and $(d^+, d_{max}]$. Different approximations are applied within each region:

1. Lower Bound Function σ_{lo} :

- $z \in [d_{min}, d^{-})$: set $\sigma_{lo}(z) = 0$.
- $z \in [d^-, 0)$: divide the interval into n subintervals. Use the tangent line at the midpoint of each subinterval for approximation.
- $z \in [0, d^+]$: divide the interval into *n* subintervals. Use line segments connecting the endpoints of each subinterval for approximation.
- $z \in (d^+, d_{max}]$: set $\sigma_{lo}(z) = \sigma(d^+) \delta$, where $\delta > 0$.
- 2. Upper Bound Function σ_{hi} :
 - $z \in [d_{min}, d^-)$: set $\sigma_{lo}(z) = \sigma(z) + \delta$, where $\delta > 0$. This ensures $\sigma_{hi}(z) > \sigma(z)$ for all $z < d^-$.
 - $z \in [d^-, 0)$: divide the interval into n subintervals. Use the tangent line at the midpoint of each subinterval for approximation.
 - $z \in [0, d^+]$: divide the interval into *n* subintervals. Use line segments connecting the endpoints of each subinterval for approximation.
 - $z \in (d^+, d_{max}]$: set $\sigma_{hi}(z) = 1$.

To minimize approximation error, we implemented a Python script to optimize the partitioning of $[d^-, d^+]$. The subintervals in $[d^-, 0)$ and $[0, d^+]$ are partitioned based on the curvature of the sigmoid function: more subintervals are allocated in regions with higher curvature.

We defer a detailed evaluation of our sigmoid approximation to Appendix C.1. We show our sigmoid approximation provides strict upper and lower bounds for the true sigmoid value and has minimal error compared with the sigmoid approximation employed in prior zkPoT works [GGJ⁺23, MR18, MZ17].

5 Implementation and Evaluation

Testbed. We use five real-world datasets of various sizes (#samples \times #features) from the UC Irvine Machine Learning Repository and Kaggle [BK96, MRC14, Pyt23, Pee24]: Adult (30,162 \times 44), Bank (45,211 \times 51), Heart (246,013 \times 121), Chess1 (524,287 \times 19), and Chess2 (1,048,575 \times 19). The two Chess datasets are random samples from a larger collection of 107 million records. Each dataset is *Z*-score normalized before training. Our ZK protocol is implemented using the ZK library for boolean circuits from EMP-toolkit [WMK16] and uses 16 threads. Experiments not directly involving ZK (i.e., those in Sections 5.1, 5.2, and 5.3) are implemented in Python. For these, we trained logistic regression models and soft-margin SVMs using the scikit-learn library, with datasets split into 75% training and 25% testing sets. The tolerance is set to None, ensuring all models are trained for exactly 1,000 epochs. All experiments are conducted on two Amazon EC-2 c7i.4xlarge instances, each with a 16-core CPU and 32 GB of RAM.

Experiment Design. We design our experiments to answer the following five key questions:

Q1. How effective are rejection sampling attacks on existing zkPoT protocols?

In Section 5.1, we present a concrete rejection sampling attack that is highly effective on prior zkPoT approaches: with only 1,000 attempts, a malicious prover can bias up to 13.5% confidently predicted data points (with probability 0–0.45 or 0.55–1).

Q2. How effectively do ϵ -vicinal models avoid this issue?

We demonstrate in Section 5.2 that the predictions of ϵ -vicinal models closely align with those of the optimal model when the predictions are moderately confident.

Q3. To compute ϵ -vicinity when the optimal model and strong convexity parameter m are unknown, we use ϵ_{reg} to bound ϵ_{real} . How tight is the gap $|\epsilon_{\text{reg}} - \epsilon_{\text{real}}|$ in practice?

Our experiments in Section 5.3 show that $|\epsilon_{reg} - \epsilon_{real}|$ is typically very small. When λ is tuned to maximize model accuracy, as is standard in machine learning, the median of these two gaps combined is less than 0.01. This indicates that both the strong convexity and regularization gaps are tight and reliable.

Q4. When computing our ϵ -vicinal bound, how tight is the overestimation gap $\Delta_{fp} = |\epsilon_{fp} - \epsilon_{reg}|$ introduced by interval arithmetic and sigmoid approximation?

As detailed in Section 5.4, $|\epsilon_{fp} - \epsilon_{reg}|$ remains minimal when these two techniques are combined, resulting in final Δ_{fp} values smaller than 8.2×10^{-3} .

Q5. In our final proof, when all three gaps are combined, is ϵ_{fp} small? Does our method offer better efficiency compared to existing *zkPoT* approaches?

In Section 5.5, we show that our protocol can prove tight ϵ -vicinity bound (with $\epsilon_{fp} < 0.01$) for logistic regression. This means our protocol guarantees that the proved model *perfectly* approximates all confident predictions from the optimal model. Our proof has up to 246× smaller Boolean circuit size compared with our baseline and is roughly $30\times$ faster than [GGJ⁺23] for 1,000 epochs. Moreover, the improvement scales linearly with the number of epochs, as our proof is independent of the number of training epochs.

5.1 Effectiveness of Rejection Sampling

As discussed before, previous zkPoT approaches are vulnerable to a rejection sampling attack. Specifically, if the prover wants to influence the prediction for a specific data point x in a binary classification task, it can locally train the model multiple times, each with a different seed. Once it finds a seed that "flips" the prediction of x, the prover adopts this seed for the zkPoT protocol.

To demonstrate the effectiveness of this attack, we designed an experiment using logistic regression on all five datasets. We first train a reference model² w_0 with an initial randomness seed r = 0. We then train 1,000 additional perturbed models, each with a unique seed r = 1, ..., 1,000. All models were trained using identical hyperparameters, including the maximum number of training epochs, regularization parameter, stopping criterion, etc. By comparing each perturbed model's predictions on the test set against those of the optimal model, we observe how changes in randomness alone can drastically affect predicted probabilities and thus influence classification outcomes.

Figure 5 illustrates the effectiveness of rejection sampling with 1,000 perturbed models on the Adult dataset. Each box corresponds to data points that are binned by their predicted probabilities from the optimal model, and the distribution of points within each box shows how 1,000 perturbed seeds shift these predictions. The vertical spread of the boxes and their overlap with shaded regions indicate that many data points initially classified as one class under the optimal model are "flipped" to the other class by certain random seeds. In other words, by simply changing the seed, the prover can bias predictions from one class to the other (or vice versa).

²For consistency with experiments in the following section, we refer to this reference model as the "optimal model".



Figure 5: The effectiveness of rejection sampling on the Adult dataset. This figure compares predictions from the optimal model to those of 1,000 perturbed models, each trained with a different seed. Each box groups data points by their predicted probabilities from the optimal model and shows how predictions vary with different seeds. Shaded regions indicate areas where predicted labels shift between classes. The results show that the attack successfully biases many high-confidence predictions from the optimal model.

Table 1 shows the results of the rejection sampling attack on all five datasets with 1,000 seeds (in the Attack columns). We observed that the prover can bias up to 13.5% of the confident predictions from the optimal model. Our experiment thus shows that the rejection sampling attack is not only feasible but also highly effective, even with a small number of random seeds.

5.2 Prediction Error of ϵ -Vicinal Model

A natural question is if a similar attack could apply when considering optimum vicinity; i.e., rather than performing rejection sampling on random seeds, a malicious prover could locally train multiple ϵ -vicinal models and select one that flips predictions as it wishes. Theorem 1, which established that model ϵ -vicinity provides an upper bound on the prediction error relative to the optimal model's predictions, suggests that this is not the case. In this section, we validate this empirically through two experiments.

We first show ϵ -vicinal models are empirically resistant to this attack: with $\epsilon = 0.01$, no confident prediction can be biased by any of the ϵ -vicinal models. We then calculate the theoretical prediction error upper bound to validate this robustness is inherent to ϵ -vicinal models.

Evaluating prediction error of ϵ **-vicinal models.** We first simulate ϵ -vicinal models and evaluate how close their predicted probabilities align with those of the optimal model. To obtain the set of ϵ -vicinal models, we randomly sample parameter vectors w_i such that each w_i lies within an ϵ -radius hypersphere around the optimal model w^* (i.e., $||w_i - w^*||_2 \le \epsilon$). As the optimal model is not computable, we approximate it by a model with negligible gradient. A λ selected through cross-validation is used across all training.

The experiments are run using three sets of ϵ -vicinal models respectively with $\epsilon = 0.01, 0.05, 0.1$. While ϵ represents worst-case scenarios where the model is far from convergence, its value for model ϵ -vicinity in practice is typically smaller than these estimates, indicating the prediction error is also much smaller than our worst-case analysis indicates (see Section 5.5 for the specific ϵ_{fp} value derived in practical settings).

Figure 6 shows the results for 1,000 ϵ -vicinal models over the Adult dataset. In these box plots, data points are grouped by their predicted probabilities from the optimal model, and each box illustrates how

Dataset	Confident Predictions (0-0.45, 0.55-1)				Unconfident Predictions (0.45–0.55)			
	Attack	$\epsilon = 0.1$	$\epsilon = 0.05$	$\epsilon = 0.01$	Attack	$\epsilon = 0.1$	$\epsilon = 0.05$	$\epsilon = 0.01$
Adult	3.67	0.95	0.00	0.00	43.48	89.28	51.30	10.14
Bank	3.32	0.76	0.01	0.00	45.50	99.47	70.37	15.87
Chess1	13.54	2.48	0.34	0.00	44.60	92.56	56.14	11.30
Chess2	13.40	3.10	0.10	0.00	44.63	93.65	53.30	11.64
Heart	1.50	0.43	0.01	0.00	48.90	99.84	70.38	15.52

Table 1: Label changes under rejection sampling attacks on prior and proposed approaches. This table presents the percentage of data points whose predicted labels differ from the optimal model under two scenarios: (i) rejection sampling attack with 1,000 random seeds and (ii) a similar attack on our approach with 1,000 ϵ -vicinal models. Data points are grouped by the optimal model's predicted probability. The results demonstrate that the attack significantly biases predictions in the prior approach, even for high-confidence cases. In contrast, ϵ -vicinal models show strong robustness, fully preserving confident predictions with $\epsilon = 0.01$.



Figure 6: Prediction error of 1,000 ϵ -vicinal models on the Adult dataset with varying ϵ . Each box groups data points by their predicted probabilities from the optimal model. Shaded regions highlight areas where predicted labels shift between classes. ϵ -vicinal models rarely flip predictions from the optimal model, except for cases where the optimal model cannot confidently classify. Notably, with $\epsilon = 0.01$, no flips occur for high-confidence predictions.

 ϵ -vicinal model predictions differ from that baseline. Table 1 shows the results across all five datasets, where the data is grouped by their predicted probabilities from the optimal model. Each column shows the percentage of data points whose label is changed by at least one ϵ -vicinal model.

Our result shows that all ϵ values preserve the correct classification for nearly all data points. Exceptions arise only for those with probabilities near the decision boundary (between 0.45 and 0.55), where even the optimal model lacks strong confidence. In other words, ϵ -vicinal models closely align with the predictions of the optimal model and prevent inadvertently flipping classes for confidently classified points.

Prediction error bound. We evaluate prediction error bounds on every dataset to confirm that our findings are not coincidental. Specifically, we use the theoretical upper bound for logistic regression, $0.25 ||\mathbf{x}||_2 \cdot \epsilon$, as derived in Corollary 1 (where K = 0.25 for the sigmoid function). Figure 7 shows our results for $\epsilon = 0.01$. We find that most data points have an error bound below 0.05. Consequently, ϵ -vicinal models do not bias predictions outside the 0.45–0.55 range even in the worst case, consistent with empirical observations (Figure 6 and Table 1). As a result, ϵ -vicinal models inherently limit adversarial bias and can eliminate



Figure 7: Prediction error bounds of ϵ -vicinal models across all test datasets. We calculate the theoretical upper bound for prediction error, $0.25 \|\boldsymbol{x}\|_2 \cdot \epsilon$, as derived in Corollary 1, for all datasets with $\epsilon = 0.01$. The prediction error measures the maximum difference between the predictions of ϵ -vicinal models and the optimal model. Most errors fall below 0.05, ensuring that ϵ -vicinal models do not bias predictions outside the 0.45–0.55 range even in the worst case.



Figure 8: Tightness of the ϵ -vicinity upper bound with varying λ values for the Adult dataset ($\epsilon = 10^{-4}$). The tightness is quantified as the gap between the upper bound ϵ_{reg} derived in Theorem 3 and the actual distance to the optimal model ϵ_{real} . This evaluation is based on 1,000 logistic regression models trained on the Adult dataset. Larger λ values result in a tighter bound, with all gaps remaining below 6×10^{-4} .

bias when ϵ is sufficiently small. Later, we show that most of our ϵ values fall within this smaller range in practice.

5.3 Bound Tightness for *ε*-Vicinal Models

We have demonstrated that ϵ -vicinal models provide strong security guarantees against rejection sampling attacks when ϵ is small. Since the exact ϵ_{real} cannot be computed, we instead calculate an upper bound ϵ_{reg} (Theorem 2). To retain the security guarantee of ϵ -vicinal models, it is crucial for the gap between ϵ_{reg} and ϵ_{real} to be sufficiently tight. In this section, we quantify how the upper bound ϵ_{reg} approximates the true distance ϵ_{real} .

We first train a reference model w^* with a large number of epochs and treat it as an estimate of the optimal model. Next, we randomly generate 1,000 ϵ -vicinal models such that $||w_i - w^*||_2 \leq \epsilon$. For each generated ϵ -vicinal model, we compute both its actual Euclidean distance from the reference model, $\epsilon_{\text{real}} = ||w_i - w^*||_2$, and the upper bound $\epsilon_{\text{reg}} = (1/\lambda) \cdot ||\nabla \mathcal{L}(w_i) - \nabla \mathcal{L}(w^*)||_2$, as derived in Theorem 3.

Dataset			Ou	Baseline			
	ϵ_{fp}	Time (hour)	Comm. (GB)	#AND (billion)	#MULT (billion)	#AND ↑	#MULT ↑
Adult	0.0073	0.22	4.28	33.87	0.47	$245.91 \times$	$4.29 \times$
Bank	0.0084	0.38	11.66	58.78	0.81	$245.71 \times$	$4.32 \times$
Chess1	0.0077	1.66	138.70	257.30	4.15	$246.06 \times$	$3.70 \times$
Chess2	0.0080	3.35	203.26	514.60	8.29	$246.04 \times$	$3,71 \times$
Heart	0.0081	4.89	106.42	755.38	9.77	$245.05\times$	$4.59 \times$

Table 2: End-to-end performance of our protocol vs. the baseline zkPoT implementation for logistic regression. We benchmark both protocols across various datasets, training for 1,000 epochs. Metrics include the proved upper bound ϵ_{fp} , execution time (**Time**), communication cost (**Comm.**), Boolean circuit size (**#AND**), and estimated arithmetic circuit size (**#MULT**). For the baseline, the values represent the factor by which the circuit size increases when proving the entire training. Communication cost is the maximum data sent by any party. Our protocol achieves significantly smaller circuit sizes, with up to $246.06 \times$ fewer AND gates and $4.59 \times$ fewer MULT gates compared to the baseline.

Although w^* is not the true optimal model, this comparison provides insight into the tightness of the bound: a tight bound between arbitrary model pairs suggests a similarly tight bound when one model is optimal. As a result, our experiment corresponds to the first two gaps $\Delta_{sc} + \Delta_{rg}$ in our roadmap in Section 1.

To determine an appropriate ϵ_{reg} for generating ϵ -vicinal models, we trained 1,000 logistic regression models on the Adult dataset for 1,000 epochs and found all ϵ_{reg} values were smaller than 10^{-5} . To account for potential precision errors due to the absence of interval arithmetic, we conservatively set $\epsilon_{\text{reg}} = 10^{-4}$. Additionally, we evaluated the impact of varying λ on the bound's tightness, as discussed in Section 3.3.

Figure 8 presents the absolute differences $\epsilon_{reg} - \epsilon_{real}$ as a box plot for various λ values with $\epsilon = 10^{-4}$. These results confirm that varying λ influences bound tightness, with all gaps remaining below 6×10^{-4} . This experiment confirms that the combined gap, $\Delta_{sc} + \Delta_{rg}$, is tightly controlled: the overall tightness error is of the same order of magnitude as ϵ_{real} .

5.4 Evaluating Fixed-Point Interval Arithmetic

Our ZK proof relies on fixed-point interval arithmetic to ensure soundness, which inherently leads to an overestimated upper bound ϵ_{fp} on ϵ -vicinity. To quantify the impact of this overestimation, we measured the resulting overestimation error Δ_{fp} when integrating sigmoid approximation into the fixed-point interval arithmetic. Because exact real-number calculations are infeasible, we used double-precision floating-point arithmetic as a reference point for comparison. The overestimation error is defined as the difference between the fixed-point upper bound computed in interval arithmetic and the floating-point "real" upper bound: $\Delta_{fp} = \epsilon_{fp} - \epsilon_{reg}$.

Specifically, we set sigmoid approximation with d = 10 (20 subintervals in total) and represented all values using fixed-point with 16 integer bits and 24 fractional bits for consistency with our final zero-knowledge implementation. After training 500 logistic regression models, we calculated their overestimation error $\Delta_{fp} = \epsilon_{fp} - \epsilon_{reg}$ introduced by fixed-point interval arithmetic and sigmoid approximation. Our result shows all values of Δ_{fp} fall between 6.6×10^{-3} and 8.2×10^{-3} (see Figure 9 in Appendix C.2 for the distribution). It confirms that while our fixed-point interval arithmetic does introduce some overestimation error, it remains small in practice.

5.5 End-to-End Performance and Comparison

Finally, we evaluate the end-to-end performance of proving model ϵ -vicinity in ZK. We compare our method with prior zkPoT protocols that verify the entire training procedure in ZK.

Experiment setup. In practice, a prover would naturally begin by training multiple models locally and choosing the one that achieves the smallest ϵ -vicinity upper bound to prove it to a verifier. We replicate this process for our end-to-end experiment: for each dataset, we performed cross-validation over $\lambda \in \{0.001, 0.01, 0.1, 1\}$, then train 1,000 logistic regression models and soft-margin SVMs for 1,000 epochs using the λ that yields the best accuracy on the test set. From these models, we select the one with the smallest ϵ -vicinity upper bound and measure the execution time, communication overhead, and circuit size to prove ϵ -vicinity in ZK. For our protocol implementation, we integrate both our fixed-point interval arithmetic and sigmoid approximation techniques into our protocol. We use fixed-point interval arithmetic (16 integer and 24 fractional bits) and a sigmoid approximation with d = 10 (20 subintervals). We compare these metrics against a baseline zkPoT protocol that verifies the entire training process. Our baseline protocol uses fixed-point numbers with 16 integer and 24 fractional bits (no interval arithmetic) and is implemented using the ZK library for Boolean circuit from emp-toolkit [WMK16]. We include the result of soft-margin SVMs in Appendix C.3.

Evaluating performance. Table 2 shows our end-to-end results. Our protocol reliably proves very small ϵ_{fp} across all datasets for both logistic regression and soft-margin SVMs. Consequently, our proofs can serve as certificates that those ϵ -vicinal models closely approximate the optimal model for most of the confident predictions: for logistic regression, all models proven have $\epsilon \leq 0.01$, which ensures that they perfectly approximate the mathematical optimal model for confident predictions; i.e., for those data points, their predictions are exactly the same as these from the optimal model. (See Section 5.2 and Table 1 for a detailed analysis of prediction accuracy under different ϵ values.)

Comparison with prior zkPoT protocols. As already highlighted throughout the paper, existing approaches to zkPoT have limited security due to their susceptibility to rejection sampling attacks. Nevertheless, we also consider how our protocols compare in terms of efficiency.

We first compare the computational cost of our protocol with that of verifying the complete training process (1,000 epochs) using the baseline zkPoT implementation. Specifically, we evaluate the Boolean and arithmetic circuit sizes, as summarized in Table 2. Our protocol achieves a circuit size reduction of up to $246 \times$ compared to the baseline zkPoT.

Next, we compare our protocol with the protocol from [GGJ⁺23]. Using their reported benchmark on a dataset with 2^{18} samples and 128 features (similar to the Heart dataset), we estimate that the total prover time for verifying 1,000 epochs of logistic regression using batch gradient descent would be approximately 145 hours, which is roughly $30 \times$ longer than our method. However, since their protocol relies on a different ZKP backend, the execution time should be viewed as a rough estimation. Additionally, we note that their protocol is optimized for proof size and RAM efficiency. We comment it is possible to integrate their ZK protocol and other existing or future optimizations into our approach to enhance our proof size and RAM usage.

Finally, we acknowledge that in some cases, it is not possible to provide a direct comparison (e.g., with Kaizen [APKP24]) due to the fact that they target neural networks, and thus models cannot be represented as solutions to convex optimization problems and our techniques cannot apply.

Acknowledgements

The work of Xiao Wang and Gefei Tan is supported by Goole Research Awards and NSF awards #2236819 and #2318975.

References

- [APKP24] Kasra Abbaszadeh, Christodoulos Pappas, Jonathan Katz, and Dimitrios Papadopoulos. Zeroknowledge proofs of training for deep neural networks. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, ACM CCS 2024, pages 4316–4330. ACM Press, October 2024.
- [Bac14] Francis Bach. Adaptivity of averaged stochastic gradient descent to local strong convexity for logistic regression. *J. Mach. Learn. Res.*, 15(1):595–627, January 2014.
- [BBB⁺18a] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In 2018 IEEE Symposium on Security and Privacy, pages 315–334. IEEE Computer Society Press, May 2018.
- [BBB⁺18b] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In 2018 IEEE Symposium on Security and Privacy (SP), pages 315–334, 2018.
- [BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 757–788. Springer, Cham, August 2018.
- [BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018.
- [BBMH⁺21] Carsten Baum, Lennart Braun, Alexander Munch-Hansen, Benoît Razet, and Peter Scholl. Appenzeller to brie: Efficient zero-knowledge proofs for mixed-mode arithmetic and Z2k. In Giovanni Vigna and Elaine Shi, editors, ACM CCS 2021, pages 192–211. ACM Press, November 2021.
- [BBMHS22] Carsten Baum, Lennart Braun, Alexander Munch-Hansen, and Peter Scholl. Moz \mathbb{Z}_{2^k} arella: Efficient vector-OLE and zero-knowledge proofs over \mathbb{Z}_{2^k} . In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 329–358. Springer, Cham, August 2022.
- [BCG⁺14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In 2014 IEEE Symposium on Security and Privacy, pages 459–474. IEEE Computer Society Press, May 2014.
- [BK96] Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

- [BMRS21] Carsten Baum, Alex J. Malozemoff, Marc B. Rosen, and Peter Scholl. Mac'n'cheese: Zeroknowledge proofs for boolean and arithmetic circuits with nested disjunctions. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 92–122, Virtual Event, August 2021. Springer, Cham.
- [BP21] Burak Bartan and Mert Pilanci. Neural spectrahedra and semidefinite lifts: Global convex optimization of polynomial activation neural networks in fully polynomial-time. *CoRR*, abs/2101.02429, 2021.
- [BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
- [CHM⁺20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Cham, May 2020.
- [CSD24] Dami Choi, Yonadav Shavit, and David Duvenaud. Tools for verifying neural models' training data. In Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23, Red Hook, NY, USA, 2024. Curran Associates Inc.
- [DILO22] Samuel Dittmer, Yuval Ishai, Steve Lu, and Rafail Ostrovsky. Improving line-point zero knowledge: Two multiplications for the price of one. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, ACM CCS 2022, pages 829–841. ACM Press, November 2022.
- [EP21a] Tolga Ergen and Mert Pilanci. Global optimality beyond two layers: Training deep relu networks via convex programs. In *International Conference on Machine Learning*, pages 2993–3003. PMLR, 2021.
- [EP21b] Tolga Ergen and Mert Pilanci. Implicit convex regularizers of {cnn} architectures: Convex optimization of two- and three-layer networks in polynomial time. In *International Conference on Learning Representations*, 2021.
- [FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987.
- [GAZ⁺22] Paul Grubbs, Arasu Arun, Ye Zhang, Joseph Bonneau, and Michael Walfish. Zero-knowledge middleboxes. In Kevin R. B. Butler and Kurt Thomas, editors, USENIX Security 2022, pages 4255–4272. USENIX Association, August 2022.
- [GGG17] Zahra Ghodsi, Tianyu Gu, and Siddharth Garg. Safetynets: verifiable execution of deep neural networks on an untrusted cloud. In *Proceedings of the 31st International Conference* on Neural Information Processing Systems, NIPS'17, page 4675–4684, Red Hook, NY, USA, 2017. Curran Associates Inc.
- [GGJ⁺23] Sanjam Garg, Aarushi Goel, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, Guru-Vamsi Policharla, and Mingyuan Wang. Experimenting with zero-knowledge proofs of training. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, ACM CCS 2023, pages 1880–1894. ACM Press, November 2023.

- [GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In 17th ACM STOC, pages 291–304. ACM Press, May 1985.
- [Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Berlin, Heidelberg, May 2016.
- [GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over Lagrange-bases for occumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019.
- [HJVE01] T. Hickey, Q. Ju, and M. H. Van Emden. Interval arithmetic: From principles to implementation. J. ACM, 48(5):1038–1068, September 2001.
- [KBPB19] John Kelsey, Luís TAN Brandão, Rene Peralta, and Harold Booth. A reference for randomness beacons: Format and protocol version 2. Technical report, National Institute of Standards and Technology, 2019.
- [Lab17] SCIPR Lab. libsnark: a c++ library for zksnark proofs., 2017.
- [LAH⁺22] Ning Luo, Timos Antonopoulos, William R. Harris, Ruzica Piskac, Eran Tromer, and Xiao Wang. Proving UNSAT in zero knowledge. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, ACM CCS 2022, pages 2203–2217. ACM Press, November 2022.
- [LKKO24] Seunghwa Lee, Hankyung Ko, Jihye Kim, and Hyunok Oh. vcnn: Verifiable convolutional neural network based on zk-snarks. *IEEE Transactions on Dependable and Secure Computing*, 21(4):4254–4270, 2024.
- [LLLX23] Qi Li, Zhuotao Liu, Qi Li, and Ke Xu. martFL: Enabling utility-driven data marketplace with a robust and verifiable federated learning architecture. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, ACM CCS 2023, pages 1496–1510. ACM Press, November 2023.
- [LWS⁺24] Ning Luo, Chenkai Weng, Jaspal Singh, Gefei Tan, Mariana Raykova, and Ruzica Piskac. Privacy-preserving regular expression matching using TNFA. In Joaquin Garcia-Alfaro, Rafał Kozik, Michał Choraś, and Sokratis Katsikas, editors, *ESORICS 2024, Part II*, volume 14983 of *LNCS*, pages 225–246. Springer, Cham, September 2024.
- [MR18] Payman Mohassel and Peter Rindal. ABY³: A mixed protocol framework for machine learning. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, ACM CCS 2018, pages 35–52. ACM Press, October 2018.
- [MRC14] S. Moro, P. Rita, and P. Cortez. Bank Marketing. UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C5K306.
- [MZ17] Payman Mohassel and Yupeng Zhang. SecureML: A system for scalable privacy-preserving machine learning. In 2017 IEEE Symposium on Security and Privacy, pages 19–38. IEEE Computer Society Press, May 2017.

- [NWH⁺14] Che Ngufor, Janusz Wojtusiak, Andrea Hooker, Talha Oz, and Jack Hadley. Extreme logistic regression: A large scale learning algorithm with application to prostate cancer mortality prediction. In *The Florida AI Research Society*, 2014.
- [PE20] Mert Pilanci and Tolga Ergen. Neural networks are convex regularizers: Exact polynomialtime convex optimization formulations for two-layer networks. In *International Conference* on Machine Learning, pages 7695–7705. PMLR, 2020.
- [Pee24] Joann Peeler. Chess engine training data. https://www.kaggle.com/datasets/ joannpeeler/labeled-chess-positions-109m-csv-format/, 2024. Kaggle.
- [Pyt23] Kamil Pytlak. 2022 Annual CDC Survey Data on Indicators of Heart Disease. https://www.kaggle.com/datasets/kamilpytlak/ personal-key-indicators-of-heart-disease, 2023. Kaggle.
- [SAB24] Megha Srivastava, Simran Arora, and Dan Boneh. Optimistic verifiable training by controlling hardware nondeterminism. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [SBLZ24] Haochen Sun, Tonghe Bai, Jason Li, and Hongyang Zhang. zkdl: Efficient zero-knowledge proofs of deep learning training. *IEEE Transactions on Information Forensics and Security*, pages 1–1, 2024.
- [SEO⁺22a] Arda Sahiner, Tolga Ergen, Batu Ozturkler, Burak Bartan, John M. Pauly, Morteza Mardani, and Mert Pilanci. Hidden convexity of wasserstein GANs: Interpretable generative models with closed-form solutions. In *International Conference on Learning Representations*, 2022.
- [SEO⁺22b] Arda Sahiner, Tolga Ergen, Batu Ozturkler, John Pauly, Morteza Mardani, and Mert Pilanci. Unraveling attention via convex duality: Analysis and interpretations of vision transformers. In *International Conference on Machine Learning*, pages 19050–19088. PMLR, 2022.
- [Soh14] Houshang Sohrab. *Basic real analysis, Second edition.* Springer New York, 01 2014.
- [SR21] César Sabater and Jan Ramon. Zero knowledge arguments for verifiable sampling. In *NeurIPS* 2021 Workshop Privacy in Machine Learning, 2021.
- [STC⁺24] Ali Shahin Shamsabadi, Gefei Tan, Tudor Ioan Cebere, Aurélien Bellet, Hamed Haddadi, Nicolas Papernot, Xiao Wang, and Adrian Weller. Confidential-dpproof: Confidential proof of differentially private training. In *International Conference on Learning Representations* (*ICLR*), 2024.
- [SWF⁺22] Ali Shahin Shamsabadi, Sierra Calanda Wyllie, Nicholas Franzese, Natalie Dullerud, Sébastien Gambs, Nicolas Papernot, Xiao Wang, and Adrian Weller. Confidential-profitt: confidential proof of fair training of trees. In *The Eleventh International Conference on Learning Representations*, 2022.
- [WMK16] Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. EMP-toolkit: Efficient multiparty computation toolkit. https://github.com/emp-toolkit, 2016.

- [WS09] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009.
- [WYX⁺21] Chenkai Weng, Kang Yang, Xiang Xie, Jonathan Katz, and Xiao Wang. Mystique: Efficient conversions for zero-knowledge proofs with applications to machine learning. In Michael Bailey and Rachel Greenstadt, editors, USENIX Security 2021, pages 501–518. USENIX Association, August 2021.
- [WYY⁺22] Chenkai Weng, Kang Yang, Zhaomin Yang, Xiang Xie, and Xiao Wang. AntMan: Interactive zero-knowledge proofs with sublinear communication. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, ACM CCS 2022, pages 2901–2914. ACM Press, November 2022.
- [XZL⁺23] Zhibo Xing, Zijian Zhang, Meng Li, Jiamou Liu, Liehuang Zhu, Giovanni Russello, and Muhammad Rizwan Asghar. Zero-knowledge proof-based practical federated learning on blockchain, 2023.
- [YSWW21] Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. QuickSilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2986–3001. ACM Press, November 2021.
- [YYH03] Hwanjo Yu, Jiong Yang, and Jiawei Han. Classifying large data sets using svms with hierarchical clusters. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, page 306–315, New York, NY, USA, 2003. Association for Computing Machinery.
- [Zho18] Xingyu Zhou. On the fenchel duality between strong convexity and lipschitz continuous gradient. *arXiv preprint arXiv:1803.06573*, 2018.

A Additional Mathematical Prelimnaries

Our methods require the prediction function to be Lipschitz continuous; thus, we provide the relevant definitions here.

A.1 Lipschitz Continuous Functions.

We now define the notion of Lipschitz continuity. This property ensures bounded changes in the function's output whenever the inputs change, and it plays a crucial role in analyzing error bounds and stability in optimization and machine learning models.

Definition 4 (Lipschitz Continuity [Soh14]). For any metric spaces (\mathbf{M}, d) and (\mathbf{M}', d') and any $X \subset \mathbf{M}$, we say that a function $f : X \to \mathbf{M}'$ is K-Lipschitz continuous if there is a constant K > 0 such that

$$d'(f(x), f(x')) \le K \cdot d(x, x') \quad \forall x, x' \in X$$

In this general form, Lipschitz continuity tells us how sensitively a function's output responds to changes in its input. In this paper, we focus on how a model's prediction $p_x = f(w, x)$ for a fixed data point x varies with different parameters w. This leads us to define a parameter-wise version of Lipschitz continuity. **Definition 5** (Parameter-wise Lipschitz Continuity). Let $f : \mathcal{W} \times \mathcal{X} \to \mathbb{R}$ be a function taking two inputs $w \in \mathcal{W}$ and $x \in \mathcal{X}$. We say that f is K-Lipschitz continuous with respect to w if there exists a constant K > 0 such that, for all $w, w' \in \mathcal{W}$ and any $x \in \mathcal{X}$,

$$|f(\boldsymbol{w}, \boldsymbol{x}) - f(\boldsymbol{w}', \boldsymbol{x})| \le K \|\boldsymbol{w} - \boldsymbol{w}'\|_2.$$

This definition is particularly useful in analyzing how small perturbations to the model parameters influence the resulting predictions on a given data point.

B Additional Proofs of Theorems

We first present a useful lemma stating that the sum of a *m*-strongly convex function and a convex function is also *m*-strongly convex.

Lemma 2. Let $f, g, h: \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable functions. Suppose f(x) = g(x) + h(x), where g is m-strongly convex and h is convex. Then f is m-strongly convex.

Proof. Since g is m-strongly convex, for any $x, y \in \mathbb{R}^n$, we have

$$g(y) \ge g(x) + \nabla g(x)^{\top} (y - x) + \frac{m}{2} ||y - x||^2$$

Meanwhile, because h is convex, we have

$$h(y) \ge h(x) + \nabla h(x)^{\top} (y - x).$$

Adding these inequalities yields

$$g(y) + h(y) \ge g(x) + h(x) + (\nabla g(x) + \nabla h(x))^{\top} (y - x) + \frac{m}{2} ||y - x||^2.$$

Therefore,

$$f(y) \ge f(x) + (\nabla f(x))^{\top} (y - x) + \frac{m}{2} ||y - x||^2,$$

since $\nabla f(x) = \nabla g(x) + \nabla h(x)$.

Hence, f is m-strongly convex.

Proof of Theorem 3 (*e*-vicinity for logistic regression).

Proof. We will show that $\mathcal{L}_{\text{logit}}^{\lambda}$ is strongly convex with parameter $m = \lambda$. Then, Theorem 3 follows from Theorem 2. First, note the logistic loss (without regularization)

$$\mathcal{L}_{\text{logit}}(\boldsymbol{w}) = -\sum_{i=1}^{n} \left[y_i \log \sigma \left(\boldsymbol{w}^{\top} \boldsymbol{x}_i \right) + (1 - y_i) \log \sigma \left(- \boldsymbol{w}^{\top} \boldsymbol{x}_i \right) \right]$$

is convex in w. Next, note the regularization term

$$R(oldsymbol{w}) = rac{\lambda}{2} \|oldsymbol{w}\|_2^2,$$

is λ -strongly convex. From Lemma 2, we conclude $\mathcal{L}_{\text{logit}}^{\lambda}$ is λ -strongly convex.

Proof of Theorem 4 (*e*-vicinity for soft-margin SVMs).

Proof. We will show $\mathcal{L}^{\lambda}_{SVM}(\boldsymbol{w}, b)$ is λ -strongly convex to \boldsymbol{w} . Then, Theorem 4 follows from from Theorem 2. First, note the hinge loss

$$\mathcal{L}_{\text{loss}}(\boldsymbol{w}, b) = \max\left(0, 1 - y_i(\boldsymbol{w}^{\top}\boldsymbol{x}_i + b)\right)$$

is convex in \boldsymbol{w} . This follows because the maximum of affine functions is convex, and $1 - y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)$ is affine in \boldsymbol{w} . Next, note the regularization term

$$R(\boldsymbol{w}) = \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2,$$

is λ -strongly convex. From Lemma 2, we conclude $\mathcal{L}^{\lambda}_{\text{SVM}}(\boldsymbol{w}, b)$ is λ -strongly convex in \boldsymbol{w} .

Proof of Theorem 5 (security of Π_{VIC-ZK}).

Proof. Corrupted Verifier. We first consider the case where the verifier is corrupted. For a corrupted verifier \mathcal{V}^* , we construct a simulator $\mathcal{S}_{\mathcal{V}}$ that calls \mathcal{V}^* as a subroutine and interacts with \mathcal{F}_{VIC-ZK} to simulates the view of \mathcal{V}^* . $\mathcal{S}_{\mathcal{V}}$ proceeds as follows:

- 1. Commit: $S_{\mathcal{V}}$ sends commit to \mathcal{F}_{VIC-ZK} .
- 2. **Prove vicinity:** $S_{\mathcal{V}}$ obtains (prove, $C_{L,\epsilon,m}$) from \mathcal{V}^* . If the circuit is incorrect, the simulator aborts; otherwise, send true to \mathcal{V}^* .

We first argue \mathcal{V}^* 's view is perfectly simulated. To see this, note that \mathcal{V}^* 's view only contains one message it receives from $\mathcal{F}_{\mathsf{ZK}}$ output $\in \{\mathsf{true}, \bot\}$. When \mathcal{V}^* behaves honestly, its view is independent of the message it sends. If it cheats, it can only do so by sending an incorrect circuit, in which case it will receive \bot in both real-world and ideal-world execution because both $\mathcal{F}_{\mathsf{ZK}}$ and $\mathcal{F}_{\mathsf{VIC}-\mathsf{ZK}}$ will abort.

Now we consider the output of an honest prover \mathcal{P} with \mathcal{V}^* . If \mathcal{V}^* behaves honestly, \mathcal{P} outputs (checked, ϵ) in both real-world and ideal-world execution. If \mathcal{V}^* cheats, \mathcal{P} output \perp in both real-world and ideal-world execution since both $\mathcal{F}_{\mathsf{ZK}}$ and $\mathcal{F}_{\mathsf{VIC}-\mathsf{ZK}}$ will abort. This completes the proof for a corrupted verifier.

Corrupted Prover. We now consider a corrupted prover \mathcal{P}^* and construct a simulator $\mathcal{S}_{\mathcal{P}}$ who invoke \mathcal{P}^* as a subroutine and interacts with \mathcal{F}_{VIC-ZK} to simulate the view of \mathcal{P}^* . The simulator $\mathcal{S}_{\mathcal{P}}$ proceeds as follows: 1. **Commit:** $\mathcal{S}_{\mathcal{P}}$ receives (commit, w') from \mathcal{P}^* , parses w' as (w, D), and record it.

2. **Prove vicinity:** $S_{\mathcal{P}}$ receives (prove, $\mathcal{C}_{\mathcal{L},\epsilon,m}$) from \mathcal{P}^* . If the circuit is incorrect or $\mathcal{C}_{\mathcal{L},\epsilon,m}(\boldsymbol{w};D) \neq 1$, the simulator aborts; otherwise, send (checked, ϵ) to \mathcal{P}^* .

We first argue the view of \mathcal{P}^* is perctly simulated by $\mathcal{S}_{\mathcal{P}}$. To see this, note \mathcal{P}^* receives (checked, ϵ) in both worlds unless it cheated, in which case it receives \bot , so the view of \mathcal{P}^* is identical in both worlds. We then only need to consider the output of an honest verifier \mathcal{V} in both executions. First note if \mathcal{P}^* behaves honestly, the output of \mathcal{V} in $\Pi_{V|C-ZK}$ and $\mathcal{F}_{V|C-ZK}$ are as follows:

- Real world: \mathcal{V} receives (checked, ϵ) iff. $\mathcal{C}_{\mathcal{L},\epsilon,m}(\boldsymbol{w},D) = 1$, i.e., $\epsilon \geq (1/m) \cdot \|\mathcal{L}_D(\boldsymbol{w})\|_2$.
- Ideal world: \mathcal{V} receives (checked, ϵ) iff. $\epsilon \geq \|\boldsymbol{w} \boldsymbol{w}^*\|_2$.

As we established in Theorem 2, $(1/m) \cdot \|\mathcal{L}_D(w)\|_2$ is an upper bound on $\|w - w^*\|_2$. Therefore, any ϵ that passes the check in the real world will also pass the check in the ideal world. Thus, the outputs of \mathcal{V} are identical in both worlds.

If \mathcal{P}^* cheats, it can only do so in the two messages it sends. We consider all three possible cases.



Figure 9: Distribution of overestimation error Δ_{fp} for the Adult dataset. Δ_{fp} is measured as the difference between the double-precision floating-point result (estimating ϵ_{reg}) and the fixed-point result ϵ_{fp} when computing ϵ -vicinity upper bound for 500 logistic regression models trained on the Adult dataset. Fixed point numbers have 16 integer bits and 24 fractional bits.

- 1. \mathcal{P}^* sends an incorrect or different circuit $\hat{\mathcal{C}} \neq C_{L,\epsilon,m}$.
- 2. \mathcal{P}^* commits to a $w' = (\hat{w}, \hat{D})$ such that $C_{L,\epsilon,m}(\hat{w}; \hat{D}) \neq 1$.

3. \mathcal{P}^* cheats by doing both 1 and 2.

In all cases, \mathcal{F}_{VIC-ZK} and \mathcal{F}_{ZK} will both abort, and \mathcal{V} outputs \perp . Therefore, the output of \mathcal{V} in both the real-world and ideal-world execution is identical. This completes the proof.

C Additional Evaluation Results

Dataset	Ours					Baseline		
	ϵ_{fp}	Time (hour)	Comm. (GB)	#AND (billion)	#MULT (billion)	#AND ↑	#MULT ↑	
Adult	0.014	0.22	4.25	33.95	0.42	$369.04 \times$	$13.79 \times$	
Bank	0.018	0.38	11.65	58.90	0.73	$368.80 \times$	$13.52 \times$	
Chess1	0.0072	1.66	138.93	258.57	3.19	$370.49 \times$	$16.15 \times$	
Chess2	0.0082	3.31	203.80	517.14	6.38	$370.47 \times$	$16.15 \times$	
Heart	0.061	4.84	106.49	755.98	9.33	$367.78 \times$	$12.68 \times$	

Table 3: End-to-end performance of our protocol vs. the baseline zkPoT implementation for soft-margin SVMs. We benchmark both protocols across various datasets, training for 1,000 epochs. Metrics include the proved upper bound ϵ_{fp} , execution time (**Time**), communication cost (**Comm.**), Boolean circuit size (**#AND**), and estimated arithmetic circuit size (**#MULT**). For the baseline, the values represent the factor by which the circuit size increases when proving the entire training. Communication cost is the maximum data sent by any party. Our protocol achieves significantly smaller circuit sizes, with up to $370.49 \times$ fewer **#AND** gates and $16.15 \times$ fewer **#MULT** gates compared to the baseline.

C.1 Sigmoid Approximation

Our sigmoid approximation (§ 4.4) is defined by two functions, σ_{lo} and σ_{hi} , which provide strict lower and upper bounds on the true sigmoid function, respectively. For our evaluation, we choose d = 10 segments (leading to a total of 20 subintervals) for consistency with our final ZK implementation. We plot σ_{lo} and σ_{hi}



Figure 10: Function values and approximation error of our lower bound function σ_{lo} , our upper bound function σ_{hi} (d = 10), and the piecewise sigmoid used in [GGJ⁺23, MR18, MZ17, APKP24].

and compare their approximation error with that of the piecewise approximation employed in prior zkPoT works [GGJ⁺23, MR18, MZ17, APKP24]. As shown in Figure 10, our method offers a tighter fit to the true sigmoid function and has a small approximation error. Moreover, the approximation error of σ_{lo} is never positive, and the error for σ_{hi} is never negative, meaning these functions serve as guaranteed lower and upper bounds for the true sigmoid.

We establish the theorem on the correctness of our sigmoid approximation in interval arithmetic.

Theorem 6 (Correctness of Sigmoid Approximation.). Let $\sigma(z)$ be the sigmoid function. Suppose there exist functions $\sigma_{lo}(z)$ and $\sigma_{hi}(z)$ such that for all $z \in \mathbb{R}$, $\sigma_{lo}(z) \leq \sigma(z) \leq \sigma_{hi}(z)$. Then, for any $z, z_{lo}, z_{hi} \in \mathbb{R}$ such that $z \in [z_{lo}, z_{hi}]$, function σ_{lo}, σ_{hi} satisfy

$$\sigma(z) \in [\sigma_{lo}(z_{lo}), \sigma_{hi}(z_{hi})].$$

Proof. The sigmoid function $\sigma(z)$ is monotonically increasing. For any $z \in [z_{lo}, z_{hi}]$, we have:

$$\sigma(z_{lo}) \le \sigma(z) \le \sigma(z_{hi}).$$

Given that the approximation functions satisfy $\sigma_{lo}(z_{lo}) \leq \sigma(z_{lo})$ and $\sigma(z_{hi}) \leq \sigma_{hi}(z_{hi})$, it follows that:

$$\sigma_{lo}(z_{lo}) \le \sigma(z) \le \sigma_{hi}(z_{hi}).$$

Therefore, for any $z \in [z_{lo}, z_{hi}]$, the sigmoid satisfies:

$$\sigma(z) \in [\sigma_{lo}(z_{lo}), \sigma_{hi}(z_{hi})].$$

C.2 Overestimation Error for Fixed-Point Interval Arithmetic

Figure 9 shows the distribution of the overestimation error $\Delta_{fp} = \epsilon_{fp} - \epsilon_{reg}$, introduced by our fixed-point interval arithmetic and sigmoid approximation. We trained 500 logistic regression models on the Adult dataset for this experiment. We computed their ϵ_{reg} using double-precision floating-point numbers and ϵ_{fp} using our fixed-point interval arithmetic and sigmoid approximation. The result shows all errors are smaller

than 8.2×10^{-3} . We noticed that the total approximation error is dominated by sigmoid approximation. This is due, in part, to our choice of a relatively small parameter d = 10 for the sigmoid approximation, which divides the sigmoid function into only 20 subintervals. Reducing the overestimation error further is possible by selecting a finer partition for the sigmoid intervals (i.e., increasing d). This will only increase the number of branching operations in ZK and will be relatively cheap.

C.3 End-to-End Performance for Soft-Margin SVMs

Table 3 presents additional benchmarks of our protocol on soft-margin SVMs. All experiments are done with the same setup described in Section 5.5.