

AZKABAN: A Zero-Knowledge Abstract Analysis for Neural Networks

Sankha Das	Lucien K. L. Ng	Yibin Yang
Georgia Institute of Technology	Georgia Institute of Technology	University of Toronto
Atlanta, USA	Atlanta, USA	Toronto, Canada
<code>sdas435@gatech.edu</code>	<code>kng68@gatech.edu</code>	<code>yibiny@ece.utoronto.ca</code>

Vladimir Kolesnikov	Teodora Baluta
Georgia Institute of Technology	Georgia Institute of Technology
Atlanta, USA	Atlanta, USA
<code>kolesnikov@gatech.edu</code>	<code>teobaluta@gatech.edu</code>

Abstract

Deep neural networks (DNNs) are increasingly used in sensitive applications, where certifying properties such as adversarial robustness and fairness is crucial. Several recent works propose DNN certification systems using zero-knowledge proofs (ZKPs)— cryptographic primitives that allow verifying certificates while maintaining confidentiality of the model. While certification algorithms typically treat the DNN as a function over reals, naively translating these algorithms into finite-precision implementations can result in unsound certification due to rounding errors. In ZKPs, this unsoundness is amplified due to a larger precision loss from fixed-point arithmetic emulated using finite fields. In this work, we highlight an overlooked gap in the soundness of prior protocols. We propose AZKABAN, a system for zero-knowledge abstract interpretation-based analysis with end-to-end soundness. We introduce operators for sound interval analysis over finite-fields, including efficient ZKP-amenable algorithms for inner-products and division, while preventing privacy leaks due to non-linear activations. We implement our system which is comprehensive in terms of supporting both feed-forward and convolutional neural networks. AZKABAN improves over the state-of-the-art ZK individual fairness certification protocol by up to two orders of magnitude in end-to-end proof time. Further, it scales to much larger models than those considered in the state-of-the-art. AZKABAN also provides, to our knowledge, the first solution for ZK robustness certification.

1 Introduction

Machine learning (ML) is increasingly deployed in high-stakes decision-making, including hiring [14], finance [5, 35] and medical diagnosis [17, 45]. While deep neural networks (DNNs) achieve remarkable accuracy, their deployment in such sensitive domains warrants additional scrutiny for bias and fairness. For instance, the COMPAS recidivism prediction tool was shown to produce systematically biased risk scores against minority defendants [4]. In safety-critical domains such as medical imaging, even imperceptible perturbations to inputs can flip a DNN’s diagnosis, motivating

formal robustness guarantees [18]. These concerns have motivated substantial work in verifying that a given DNN satisfies properties such as adversarial robustness [20, 34, 50, 51, 54] and fairness [6].

Almost all existing verification algorithms, however, require white-box access to the DNN. Since ML models are typically intellectual property of companies, public certification of the fairness or robustness of a deployed model is challenging. It is thus important to develop a solution where an auditor checks that a property holds while simultaneously preserving the confidentiality of the model.

Recent works address this tension between transparency and confidentiality via the cryptographic primitive of zero-knowledge proofs (ZKP) [19, 55, 59] — protocols where a prover convinces a verifier that a statement about the prover’s secret is true while revealing nothing beyond the statement’s validity. For instance, in zero-knowledge local individual fairness certification [55], the prover proves the statement that its model (the prover’s secret) satisfies individual fairness at a queried input x^* with a fairness score δ .

In this paper, we identify a class of vulnerabilities present in two existing ZK fairness certification protocols, FairProof [55] and FairZK [59]. These vulnerabilities arise from the gap between cleartext certification’s real-valued semantics and its finite-precision realization. Cleartext certification algorithms are proven sound with respect to the infinite-precision real domain, but these algorithms are implemented on finite-precision computers, where properties such as closure and associativity do not hold for all operations. Finite-precision thus warrants rounding the intermediate computations up or down. Previously, such gaps in floating-point arithmetic have been exploited in cleartext implementations of differential privacy noise sampling [31, 42] and adversarial robustness certification [30, 32]. These soundness issues are further amplified in cryptographic ML auditing using ZKPs. Since efficient ZKP protocols approximate real arithmetic using fixed-point arithmetic that is typically much less precise than floating-point arithmetic, these exploitable rounding errors can accumulate much faster. Consequently, a malicious prover can adversarially choose its parameters to drive the rounding errors in its favor and convince the verifier into accepting false certificates. Note that this issue lies outside the ZKP layer: even if the ZKP protocol itself is sound, the underlying fixed-point implementation of the cleartext algorithm is not. To the best of our knowledge, we are the first to identify such vulnerabilities in ZKPs for fairness certification.

Beyond patches for existing fairness certification of fixed-point algorithms, we develop a set of operations for sound fixed-point arithmetic, capturing both rounding-up or rounding-down errors. Our operations leverage the mathematical framework of interval analysis [41, 43], while co-designing several novel optimizations with our ZKP setting. These include inner-products over interval vectors with lazy truncations and a specialized division algorithm using tangent and secant linear approximations to leverage efficient lookup-table based ZK non-linear function evaluations [25]. We prove that our fixed-point arithmetic is sound. We believe this contribution is a useful foundation for future DNN ZK certification.

We further develop an end-to-end ZK DNN certification system, AZKABAN¹, based on abstract interpretation [13] and our sound fixed-point arithmetic. While existing ZK fairness certification protocols struggle to achieve a good balance between scalability and precision [55, 59], abstract interpretation-based DNN certification is a scalable-yet-precise approach [20, 47, 48]. Abstract interpretation approaches construct a set of convex constraints that soundly over-approximate the DNN’s reachable outputs over all inputs within the neighborhood of a queried x^* . Among these, DeepZono [47] is particularly amenable to cryptographic protocols due to its sparse number of linear

¹A Zero-Knowledge ABstract Analysis for Neural networks.

constraints, while achieving high precision. AZKABAN empirically demonstrates ZK certification on 5 datasets and 12 diverse models for local individual fairness and robustness [49].

Summary of Contributions. Our major contributions include:

- **Unsoundness in ZK Fairness.** We show that naively implementing real-valued certification algorithms with fixed-point representation introduces rounding errors that can accumulate over certification steps and model layers. We study two recent ZK fairness protocols [55, 59], giving concrete examples of a malicious prover successfully certifying false fairness values, with accumulated errors reaching upto 0.49% even in simplified attack settings (Table 1), and propose fixes for such vulnerabilities (Section 5).
- **Sound Interval Arithmetic.** We co-design efficient operators for fixed-point intervals alongside crypto-friendly optimizations and prove their soundness under fixed-point arithmetic. Some of underlying key technical contributions include efficient algorithms for table-lookup-based division over fixed-point intervals and the use of lazy truncations for inner products (Section 6).
- **DNN ZK Certification.** We design AZKABAN, a system for zero-knowledge DNN abstract interpretation using zonotopes [47] over field intervals. We highlight several challenges in balancing zero-knowledge with proof costs, especially for non-linear activations in the zonotope domain and proving inner products of interval vectors. Our system achieves end-to-end soundness guarantees for DNN certification. To our knowledge, AZKABAN is the first solution for ZK adversarial robustness certification with end-to-end soundness over finite fields (Section 7).
- **Implementation and Evaluation.** We evaluate our system for zero-knowledge DNN certification of robustness and individual fairness with support for multi-layer perceptrons and convolutional neural networks. We also improve over the state-of-the-art protocol for ZK local individual fairness certification by 40–110 $\times$ in total proof time, while maintaining certification soundness (Section 8). Our code is open sourced at the repo <https://github.com/sankha555/azkaban>.

2 Preliminaries

We begin by setting notation that we use in the rest of the paper. We then review relevant background in neural network abstract interpretation, fixed-point representation and zero-knowledge proofs.

2.1 Notation

Let $\mathbb{R}$, $\mathbb{Z}$ and $\mathbb{N}$ denote the set of reals, integers and natural numbers, respectively. The finite-field $(\{0, 1, \dots, p-1\}, +, \cdot)$ is denoted by $\mathbb{F}_p$, where $+$ and $\cdot$ denote addition and multiplication modulo a prime p . $a \triangleq b$ denotes that a “is defined to” be b . We denote the set $\{1, \dots, n\} \subset \mathbb{N}$ by $[n]$. κ denotes the computational security parameter (e.g., 128) and σ denotes the statistical security parameter (e.g., 40). The prover is denoted by $\mathcal{P}$ and is referred to by she, her, etc. The verifier is denoted by $\mathcal{V}$ and is referred to by he, him, etc. Vectors are denoted by boldface letters (e.g., $\mathbf{a}$, $\mathbf{A}$) and round-brackets $(a_1, \dots, a_n)$. The i -th element of $\mathbf{a}$ is denoted by a_i . $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes the inner-product between two vectors $\mathbf{x}$ and $\mathbf{y}$ of the same length. x denotes that the field-element x is authenticated between $\mathcal{P}$ and $\mathcal{V}$ by an information-theoretic MAC (cf. Section 2.4) over the prime field $\mathbb{F}_p$ (for some given prime p).

2.2 Abstract Interpretation for DNNs

Abstract interpretation [13] based DNN certification algorithms enable scalable and precise certification of properties [20, 47, 48]. Such algorithms approximate the primitive computations of a DNN, corresponding to affine pre-activation computation and non-linear activations such as ReLUs and sigmoid. The building blocks are abstract transformers $T_f^\#$ for each computation f on an *abstract domain*. Abstract domains are composed of *abstract elements* χ that overapproximate the *concrete* values in the DNN’s computations.

A key property of abstract interpretation is soundness. Given a concretization function $\gamma : \{\chi\} \rightarrow X$ that maps abstract elements to concrete values and a function f that has domain X , the corresponding abstract transformer $T_f^\#(\chi)$ is **sound** iff $\{f(x) \mid x \in X\} \subseteq \gamma(T_f^\#(\chi))$. In other words, the output of a sound abstract transformer $T_f^\#$ over-approximates the concrete values output by the function f . Abstract interpreters thus soundly over-approximate an exponentially large number of concrete inputs with a single input abstract element. The different abstract transformers of the DNN compose together to transform the input abstract element χ^i into an output abstract element χ^o . The inclusion of χ^o in a safe region S_ψ certifies the input as safe w.r.t. property ψ of the model.

Abstract transformers must achieve good tradeoffs between precision and speed of certification. Since abstract elements over-approximate the DNN’s concrete values for soundness, they can contain values which the DNN never computes concretely. For example, even though the ReLU function will always output non-negative values, a sound abstract ReLU transformer can output abstract elements that over-approximate negative numbers. *Precision* of the abstract transformers measures the how tightly the over-approximated set bounds the set of values computed by the DNN. If the abstract transformers are too imprecise, the interpreter can fail to certify certain inputs as safe. Hence, such abstract interpretation-based algorithms are *sound but incomplete*, i.e., no false properties can be certified but not all true properties will be certified.

Certification accuracy measures the fraction of inputs from a test dataset which the abstract analysis certifies safe². Typical abstract interpreters for DNN certification [20, 47, 47] sacrifice some precision, and hence accuracy, for *speed* and *scalability*. Thus, there exists a tradeoff between certification speed and the certification accuracy. This tradeoff is decided by the choice of abstract domain.

Zonotopes [21] are a class of center-symmetric convex regions that give rise to abstract domains and transformers with attractive speed-accuracy tradeoffs. Our proposed algorithm AZKABAN is based on DeepZono [47], a DNN abstract analysis that uses the zonotope abstraction and preserves soundness over floating-point arithmetic. We illustrate the abstract transformers in DeepZono alongside the design of our ZK abstract interpreter in Section 7.

2.3 Fixed-Point Numbers and Intervals

Traditionally, DNNs operate over real-valued inputs and parameters represented as floating-point numbers (e.g., IEEE-754 [3]). However, typical ZKP protocols operate over finite-fields that cannot emulate floating-point arithmetic efficiently. Therefore, many cryptographic protocols use fixed-point arithmetic to operate over real numbers that are approximated to a suitable degree of precision. Despite the inherent loss in numerical precision, fixed-point representation has been widely adopted

²Note that certification accuracy is distinct from classification accuracy of a DNN which measures the correctness of its predictions.

in several secure ML protocols [23, 24, 29] due to efficiency reasons. As we show experimentally, the precision loss does not significantly affect our certification accuracy (Section 8).

To approximate a real number x in signed fixed-point representation, x is first scaled by the fixed-point scale s as $x \cdot 2^s$. Since real numbers have infinite fractional precision, $x \cdot 2^s \in \mathbb{R}$ is not necessarily an integer. Hence, the fixed-point integer representing x is obtained by rounding $x \cdot 2^s$ to its floor or ceiling value. In tasks like (secure) ML training or inference, either of the two rounding modes can be used since such applications primarily concern with maintaining proximity to the real value x .

In cryptographic protocols for ML usecases [23–25, 29, 53], fixed-point integers offer a convenient mapping from real-valued parameters/inputs to field elements. For a given prime p , the set of signed fixed-point numbers in the range $[-\frac{p-1}{2}, \frac{p-1}{2}] \subset \mathbb{Z}$ are embedded in the finite field $\mathbb{F}_p$ as follows. Every negative fixed-point integer $x \in [-\frac{p-1}{2}, -1]$ is mapped to the field element $p + x \in \mathbb{F}_p$; every non-negative fixed-point integer $x \in [0, \frac{p-1}{2}]$ is mapped to the field element $x \in \mathbb{F}_p$. The inverse mapping from $\mathbb{F}_p$ to fixed-point integers follows analogously.

We assume model parameters and inputs to be represented as fixed-point integers with a public scale s . This quantized model is deployed for inference and is of interest to the auditor for DNN certification. We later use the above mapping between fixed-point integers and $\mathbb{F}_p$ to define *field intervals* that form the basis of our sound zero-knowledge abstract interpreter.

2.4 VOLE-Based Zero-Knowledge Proofs

Zero-Knowledge Proofs (ZKPs) [22] enable a prover $\mathcal{P}$ to prove to a verifier $\mathcal{V}$ that a statement x is true without revealing any information about x beyond its validity. More formally, a statement x is said to be true with respect to a public relation $\mathcal{R}$ if there exists a witness w such that $(x, w) \in \mathcal{R}$. $\mathcal{P}$ produces a proof π about x 's validity which is verified by $\mathcal{V}$, i.e., $\mathcal{V}$ on input x and the proof π outputs 1 iff x is valid. A ZKP for the relation $\mathcal{R}$ must satisfy:

- **Completeness:** Proofs for all valid statements must be verified, i.e., $\forall (x, w) \in \mathcal{R} : \pi \leftarrow \mathcal{P}(x, w) \implies \mathcal{V}(\pi, x) = 1$.
- **Soundness:** Proofs for only valid statements must be verified, i.e., $\forall x, \pi : \exists w : (x, w) \in \mathcal{R} \implies \mathcal{V}(\pi, x) = 1$.
- **Zero-Knowledge:** The proof π must not reveal any information about w to the verifier beyond what he already knows.

Recently, there has been much interest for interactive VOLE-based ZKPs [15, 52, 56] due to their fast end-to-end proof time and low memory footprint as compared to non-interactive proofs such as SNARKs. VOLE-based ZKPs are typically designed in the commit-and-prove ZK paradigm [10] where information-theoretic MACs (IT-MACs) [8, 44] are used to commit to the witness over a finite field $\mathbb{F}_p$. Specifically, $\mathcal{V}$ holds a uniformly random global local key $\Delta_{\mathcal{V}} \in \mathbb{F}_p$ that he keeps secret throughout. For an arbitrary witness $w \in \mathbb{F}_p$ known to $\mathcal{P}$, $\mathcal{P}$ and $\mathcal{V}$ authenticate w over the finite field. Specifically, $\mathcal{P}$ obtains a MAC $M_w \in \mathbb{F}_p$ and $\mathcal{V}$ obtains a uniformly random local key $K_w \in \mathbb{F}_p$ such that $M_w = K_w + \Delta_{\mathcal{V}} \cdot w$. These commitments are both hiding and binding and satisfy additive homomorphism.

3 Problem Formulation

In this section, we motivate the problem of ZK DNN verification and specify the design goals underlying our system.

3.1 Motivation

Consider a company that provides black-box inference service for a proprietary model M . Users can query the model, typically hosted on a cloud service, on their public inputs x but should remain oblivious of the model’s parameters and intermediate computations. To enable trust in the model, users or a third-party would want to audit whether the model satisfies certain desirable properties.

One such desirable property is local adversarial robustness which we use to illustrate our problem setting. We say the model M is δ -robust for an input x if for all inputs x' within a radius- δ ℓ_∞ ball centered at x , $M(x) = M(x')$. The company publishes this radius δ (the “robustness certificate”), where a larger δ implies greater robustness to adversarial perturbations. The desired perturbation radius δ^* depends on the application, dataset and model architecture [11]. Among others, abstract interpretation based certification algorithms soundly certify if a model is δ -robust for a given input with high precision and scalability to large models.

However, two orthogonal problems arise here. First, the company can cheat by over-reporting the true robustness certificate δ as computed by abstract interpretation. Second, the company wants to preserve the confidentiality of its proprietary parameters from the users or a third-party auditor. To simultaneously address the two problems, the company must prove honest computation of δ by presenting a **zero-knowledge proof of abstract interpretation**. Note that such a proof is distinct from a proof of inference [40, 46, 53]. The latter only establishes correctness of the model’s output on a given input, not its robustness to other adversarial inputs.

3.2 Problem Statement

We address the above problem by constructing a zero-knowledge abstract interpretation system for certifying neural networks. We use DeepZono [47] as the underlying cleartext DNN abstract interpretation algorithm and carefully translate its steps into an interactive ZKP between the company (the prover $\mathcal{P}$) and the auditor (the verifier $\mathcal{V}$). We aim to achieve the following properties:

- **Certification Soundness and Accuracy:** Preserve certification soundness over fixed-point intervals and achieve high certification accuracy.
- **Model Privacy:** $\mathcal{V}$ does not learn any additional information about the neural network beyond what is known.
- **Efficiency:** The end-to-end proof costs, i.e., proof time and communication, must be concretely small.

Threat Model. We construct our system in the designated-verifier interactive ZKP setting. While the neural network parameters are private to $\mathcal{P}$, its architecture, input/output domains and activation functions are publicly known.³ $\mathcal{V}$ ’s inputs to the model are public. It is assumed that $\mathcal{P}$ and $\mathcal{V}$ communicate via secure channels that are resistant to tampering.

³This nominal leakage still preserves the majority of the proprietary value of the models and is commonplace in secure ML frameworks [23, 24, 29, 53].

4 Technical Overview

Before explaining the details of our ZK abstract interpretation system, we present a brief overview of our findings and techniques.

First, we study how existing ZK fairness certification protocols do not maintain certification soundness with respect to real numbers. Specifically, we study two protocols — FairProof [55] and FairZK [59] — that certify model fairness by proving honest computation of projected distances and statistical parity [16], respectively. However, the respective cleartext certification algorithms are proven sound only in the infinite-precision real domain. Thus, naively translating these algorithms into a ZKP over fixed-point integers results in errors from rounding, particularly for division and multiplication. Using numerical examples, we first show how a malicious prover can exploit the direction of rounding in fixed-point arithmetic to overestimate division and underestimate products that quantify model fairness. We also highlight that such errors can accumulate over multiple layers or certification steps, thus enabling the prover to certify false fairness certificates for deep neural networks. We propose fixes to existing implementations by recommending the use of adaptive rounding that selects the direction of rounding depending on the fairness metric and primitive operations therein.

Since cryptographic protocols typically operate over finite fields, emulation of floating point arithmetic is cost-inefficient for ZK DNN certification. Taking inspiration from the above fixes, we thus propose fixed-point interval analysis to perform sound DNN abstract interpretation in zero-knowledge. We co-design fixed-point interval operations with crypto-friendly optimizations such as lazy truncations in inner products and use of LUT-based sound division. Using the monotonicity of the truncation operation, lazy truncations reduce costly truncations inside zero-knowledge to one truncation per inner product. Further, our division operation relies on tangent and secant linear approximations of the reciprocal functions, leveraging recent advances in LUT-based ZKPs in the subsequent construction of our ZK abstract interpretation system.

Finally, we construct our sound ZK abstract interpretation system using the above fixed-point interval analysis. Our system is based on the DeepZono abstract interpretation algorithm, which uses the zonotope abstract domain. DeepZono is particularly useful for cryptographic certification protocols because: 1) the abstract transformers largely rely on linear constraints and 2) the number of constraints required to achieve high precision certification is far less than other abstract domains [48]. Specifically, we prove honest computation of the abstract transformers from DeepZono for different computations of the DNN, e.g. affine pre-activation and non-linear activations, using interactive IT-MAC-based ZKPs.

Two major proof overheads arise in our design. First, due to expensive comparisons required in sound and precise interval inner-products, we face challenges in vectorizing inner-product proofs that otherwise leverage fast ZKPs over field vectors [56]. Second, due to imprecision of the ReLU abstract transformers in DeepZono, we need to pad abstract elements of ReLU variables to prevent leaking information about the underlying neuron’s state; empirically, this padding adds $\approx 8\times$ overhead to the proof costs but is important to preserve model privacy. Although both challenges add significant overheads to our ZKP, AZKABAN satisfies end-to-end certification soundness in ZK, while scaling to multi-layer perceptrons and convolutional neural networks of up to thousands of neurons.

5 Unsoundness in Finite-Precision Certification

In this section, we illustrate how rounding errors in finite-precision arithmetic, specifically fixed-point arithmetic, can be exploited to over-claim the properties of a model by a malicious prover. We consider two different strategies that the adversary can adopt to introduce rounding errors in the computation of the certificate.

5.1 Direction of Approximation Matters

We first study how DNN properties can be overclaimed by exploiting the direction of rounding in fixed-point operations like division and square root. We demonstrate this unsoundness by illustrating the fixed-point operations in the FairProof protocol [55] for certifying local individual fairness. FairProof abstracts the DNN as a polyhedral complex that partitions the input space into decision regions that are separated from each other by boundary facets. It quantifies fairness by computing a projected distance of the input $\mathbf{x}^*$ from the nearest boundary facet given by the formula $d = |\mathbf{a} \cdot \mathbf{x}^* + b| / \|\mathbf{a}\|_2$, where the hyperplane containing the boundary facet is $y = \mathbf{a} \cdot x + b$. A larger distance implies greater fairness.

Fixed-point division between two fixed-point integers u and v (with scale s) is computed as $\lfloor \frac{u \cdot 2^s}{v} \rfloor$, where $\lfloor \cdot \rfloor$ indicates either rounding down or rounding up. We show that a malicious prover can falsely certify a fairness certificate that is larger than the real fairness value by exploiting a consistent rounding direction.

Fairness Overestimation by Rounding Up. Consider the case where the certification rounds up all fixed-point integers. Thus, the distance computation takes as input fixed-point integers $\{\bar{a}_i = \lceil a_i \cdot 2^s \rceil\}_{i \in [n]}, \bar{b} = \lceil b \cdot 2^s \rceil$ from the prover and $\{\bar{x}_i = \lceil x_i \cdot 2^s \rceil\}_{i \in [n]}$ from the verifier. The prover computes the projected distance in fixed-point⁴ as $D = \left\lceil \frac{|\sum_i \bar{a}_i \cdot \bar{x}_i + \bar{b} \cdot 2^s|}{\left\lceil \sqrt{\sum_i \bar{a}_i^2} \right\rceil} \right\rceil$. Then, the computed distance always exceeds the true distance if $\forall i \in [n] : a_i \cdot 2^s \in \mathbf{Z}$ and $\sum \bar{a}_i^2$ is a perfect square. While we provide a formal proof in Appendix C, this follows from the fact that the square root in the denominator is a perfect integer with a scale of exactly s and thus there are no errors due to rounding. The final ceiling operation following the division ensures that the fixed-point distance exceeds the real projected distance. Thus, the prover can prove a larger fairness certificate that exceeds the real certificate. Importantly, observe that these conditions depend only on the prover’s parameters. Hence, she can adversarially choose parameters that satisfy these conditions to over-claim the fairness for all verifier inputs. We illustrate the attack numerically below.

Consider the prover selects $\mathbf{a} = (3, 4)$, $b = 0.0977$, the verifier inputs $\mathbf{x}^* = (1.001, 1.001)$ and that $s = 10$. Then the real projected distance $d_{\mathbb{R}} = |3 \cdot 1.001 + 4 \cdot 1.001 + 0.0977| / \sqrt{9 + 16} = 1.42094$. Observe that $a_1 \cdot 2^s, a_2 \cdot 2^s \in \mathbf{Z}$ and $\sum \lceil a_i \cdot 2^s \rceil^2 = (5120)^2$. Then, $D = \lceil \bar{\mathbf{a}} \cdot \bar{\mathbf{x}} + \bar{b} \cdot 2^s \rceil / \lceil \sqrt{\sum \bar{a}_i^2} \rceil = \lceil 7457792 / 5120 \rceil = \lceil 1456.6 \rceil = 1457$. Scaling this value back to reals, the verifier accepts the fairness value of $d = 1.42285156$. Thus, the computed fixed-point distance exceeds the real distance in the third decimal place, which is larger than the least representable number (≈ 0.00098) for precision $s = 10$.

Fairness Overestimation by Rounding Down. Alternatively, the prover and verifier can agree

⁴Note that the numerator has scale $2s$ and the denominator (after the square root) has scale s . Also, square root is computed by simply finding the integer (having scale s) that when squared is closest to input (having scale $2s$) to the square-root function. Hence, these scales are consistent with fixed-point division.

upon computing fixed-point operations while rounding down non-integral values. Then, the prover computes the projected distance in fixed-point as $D = \left\lfloor \left| \sum_i \underline{a}_i \cdot \underline{x}_i + \underline{b} \cdot 2^s \right| / \left\lfloor \sqrt{\sum \underline{a}_i^2} \right\rfloor \right\rfloor$ (where, $\underline{t}$ denotes $\lfloor t \cdot 2^s \rfloor$ for $t \in \mathbb{R}$) and publishes it as the fairness certificate to the verifier. We illustrate an attack by a numerical example in Appendix C that there exist model parameters and input values under which the above fixed-point quotient exceeds the true real-valued quotient. As a result, the prover can claim a larger fairness certificate than the true fairness.

Soundly Reporting the Distance. To ensure that the distance computed in fixed-point never overestimates the real distance, we stress on the use of *adaptive rounding* rather than a consistent rounding direction. In particular, the analysis must compute fixed-point integers using *both* rounding modes. Using the fixed-point interval representation defined in Section 2.3, the distance should be

computed as $\underline{d} = \left\lfloor \frac{|\sum \underline{a}_i \cdot \underline{x}_i + \underline{b} \cdot 2^s|}{\left\lfloor \sqrt{\sum \underline{a}_i^2} \right\rfloor} \right\rfloor$. Thus, $\underline{d}$ underestimates the true (scaled) distance $d_{\mathbb{R}}$ because

of the following inequality chain over reals: $\underline{d} \leq \frac{|\sum \underline{a}_i \cdot \underline{x}_i + \underline{b} \cdot 2^s|}{\left\lfloor \sqrt{\sum \underline{a}_i^2} \right\rfloor} \leq \frac{|\sum \underline{a}_i \cdot \underline{x}_i + \underline{b} \cdot 2^s|}{\sqrt{\sum \underline{a}_i^2}} \implies \underline{d} \leq \frac{|\sum \underline{a}_i \cdot \underline{x}_i + \underline{b} \cdot 2^s|}{\sqrt{\sum \underline{a}_i^2} \cdot 2^{2s}} \leq \frac{|\sum \underline{a}_i \cdot \underline{x}_i + \underline{b}|}{\sqrt{\sum \underline{a}_i^2}} \cdot 2^s = d_{\mathbb{R}} \cdot 2^s$. By using this convention for computing distances using sound fixed-point arithmetic that considers both rounding directions, the reported fairness stays below the true fairness of the DNN.

5.2 Accumulation of Unsoundness

We now consider another strategy for over-claiming the model’s properties by relying on fixed-point rounding errors that accumulate over multiple sequential operations. To illustrate this strategy, we use the example of the FairZK [59] protocol. In contrast to FairProof that certifies individual fairness, FairZK proves correct upper bounds on statistical parity [16] to quantify group fairness of the DNN. FairZK estimates the statistical parity δ_h^H of a neural network with H layers and ReLU activations using the recursive inequalities

$$\|\delta_h^\ell\|_2 \leq L \|W^{\ell-1}\| \|\delta_h^{\ell-1}\|_2 + 2L \|\Delta_z^\ell\|_2 \text{ and } \Delta_z^\ell \leq \|W^{\ell-1}\| \cdot \Delta_z^{\ell-1},$$

where $\delta_h^0 = \delta_x$ and $\Delta_z^1 \leq \|W^0\| \cdot \Delta_x$. Here, L represents the Lipschitz constant of the ReLU function, W^ℓ represents the weight matrix of the ℓ -th layer and δ_x, Δ_x represent statistical information about the population. $\|\cdot\|$ represents the spectral norm of the matrix. Since statistical parity quantifies the difference in likelihood of outputting a favorable label for each value of a binary sensitive attribute, a smaller statistical parity implies that the model’s output is roughly equally favourable for both values. Thus, the prover *underestimates* the value of $\|\delta_h^H\|_2$ in order to over-claim the true fairness.

Recall that multiplication in fixed-point arithmetic involves truncation (by 2^{-s}) to restore the scale of the product $\gamma = \alpha \cdot \beta$, where α and β are fixed-point integers denoting $\lfloor a \cdot 2^s \rfloor$ and $\lfloor b \cdot 2^s \rfloor$ for $a, b \in \mathbb{R}$. However, the truncation result $\gamma \cdot 2^{-s} \in \mathbb{R}$ need not be an integer and hence requires rounding to an integer. If the truncation result is always *rounded down* to the nearest integer, note that the final product $\underline{\gamma} = \lfloor \gamma \cdot 2^{-s} \rfloor \leq \gamma \cdot 2^{-s} = \lfloor a \cdot 2^s \rfloor \cdot \lfloor b \cdot 2^s \rfloor \cdot 2^{-s} \leq (a \cdot b) \cdot 2^s$, thus introducing a negative error from the real product. More importantly, since the statistical parity for the final layer is computed by multiplying terms recursively layer-by-layer, this error keeps accumulating and significantly underestimates the true bound on the statistical parity. We illustrate this strategy below. The prover selects model parameters such that $\forall \ell \in [6], \|W^\ell\| = 1.125$. The statistics about the data

(which the prover does not control) are such that $\|\delta_h^0\|_2 = 1.125$ and $\Delta_z^\ell = \mathbf{0}$ for all ℓ and $L = 1$. Thus, $\|\delta_h^\ell\|_2 \leq \|W^{\ell-1}\| \|\delta_h^{\ell-1}\|_2$ for $\ell \in \{1, 2, 3, 4, 5, 6\}$. δ_h^ℓ is computed layer-by-layer as (below, we factor out the multiplication of L which is scaled to 256):

Initial δ_h^0

Real: $\delta_h^0 = 1.125$; Fixed-Point: $\delta_h^0 = \lfloor 1.125 \times 256 \rfloor = 288$

Layer 1

Real: $\|W^0\| = 1.125$, $\delta_h^0 = 1.125 \implies \delta_h^1 = 1.265625$

Fixed-Point: $\|W^0\| = \delta_h^0 = 288 \implies \delta_h^1 = \lfloor \frac{288 \times 288}{256} \rfloor = 324$

Layer 2

Real: $\|W^1\| = 1.125$, $\delta_h^1 = 1.265625 \implies \delta_h^2 = 1.423828$

Fixed-Point: $\|W^1\| = 288$, $\delta_h^1 = 324 \implies \delta_h^2 = \lfloor \frac{288 \times 324}{256} \rfloor = 364$

The analysis continues similarly for consequent layers and is tabulated in Table 1 where the column ‘‘Adv. Bound’’ presents the (scaled-down) bound computed by the prover above in fixed-point arithmetic. For the complete expression with non-zero Δ_z^ℓ , the error accumulation will be even larger. This way, a malicious prover can significantly underestimate the bound on SP and claim better fairness certificates than the real fairness level of the model.

Table 1. Error in Upper Bound of SP due to Rounding

Layer (ℓ)	Real Bound	Adv. Bound	Abs. Error	% Error
1	1.265625	1.265625	0.000000	0.00%
2	1.423828	1.421875	0.001953	0.13%
3	1.601807	1.597656	0.004150	0.26%
4	1.802032	1.796875	0.005157	0.29%
5	2.027286	2.019531	0.007755	0.38%
6	2.280697	2.269531	0.011167	0.49%

Although the bound computed in FairZK is already far looser than the theoretical upper bound of statistical parity (i.e., 1), these observations are important to avoid underestimating true statistical measures due to rounding errors in finite-bitwidth arithmetic. A straightforward patch to avoid this underestimation is to always round-up after fixed-point truncations. This way the computed statistic is never underestimated and provides a sound upper bound with respect to the real bound.

Remark. The underlying cleartext certification algorithms above have been proven sound over the (infinite-precision) real domain [33, 37]. The unsoundness we discuss stems from poor accounting of rounding errors when the algorithms are translated to the fixed-point domain — not from the zero-knowledge proof system.

6 Sound Fixed-Point Interval Analysis

Motivated by the soundness gaps identified above, we stress on the use of adaptive rounding for sound interval arithmetic. In this section, we propose sound interval arithmetic over fixed-point intervals to facilitate abstract interpretation for DNNs quantized in fixed-point representation. Our ideas draw from similar results in general [43] and floating-point interval analysis [41] but observe

certain points of efficiency in the fixed-point domain due to closure and associativity of operations that floating-point lacks.

6.1 Fixed-Point Intervals

A *fixed-point interval* $I = [\ell, u]$ is defined by a pair of interval endpoints $\ell, u \in \mathbb{Z}$ where $\ell \leq u$. For a real number x , the fixed-point interval $I = [\ell, u]$ *soundly over-approximates* x iff $\ell \leq x \cdot 2^s \leq u$.

For $x \in \mathbb{R}$, we define the *lower-bounding* fixed-point endpoint $\underline{x} \triangleq \lfloor x \cdot 2^s \rfloor$ and the *upper-bounding* fixed-point endpoint $\bar{x} \triangleq \lceil x \cdot 2^s \rceil$. Taken together, $\underline{x}$ and $\bar{x}$ define a fixed-point interval $X \triangleq [\underline{x}, \bar{x}]$ that *soundly over-approximates* x . Soundness follows from the fact that $\underline{x} = \lfloor x \cdot 2^s \rfloor \leq x \cdot 2^s \leq \lceil x \cdot 2^s \rceil = \bar{x}$. We further define a fixed-point interval $[\underline{x}, \bar{x}]$ to be *purely positive* iff $\underline{x} \geq 0$, to be *purely negative* iff $\bar{x} \leq 0$ and to be *spanning zero* iff $\underline{x} < 0$ and $\bar{x} > 0$.

We now specify operations such as addition, subtraction, multiplication and division on fixed-point intervals. For each operation, we consider two fixed-point interval operands $A = [\underline{a}, \bar{a}]$ and $B = [\underline{b}, \bar{b}]$, which overapproximate the real values a and b respectively. The result of each operation is an interval $C = [\underline{c}, \bar{c}]$. *The operation is said to be sound* if C over-approximates the result c of the operation between a and b over $\mathbb{R}$.

6.2 Addition and Subtraction

Since fixed-point numbers are integers which are closed under addition and subtraction, fixed-point interval addition (subtraction) is performed by simply adding (subtracting) the endpoints of the operand intervals with each other. Specifically, fixed-point interval addition over A and B is defined as

$$C = A \oplus B \triangleq [\underline{a} + \underline{b}, \bar{a} + \bar{b}] \quad (1)$$

Similarly, subtraction is defined as

$$C = A \ominus B \triangleq [\underline{a} - \bar{b}, \bar{a} - \underline{b}] \quad (2)$$

Lemma 1. *The operators $\oplus$ and $\ominus$ are sound.*

Note that since floating-point numbers are not closed under addition or subtraction, additional steps are required to explicitly set the rounding mode to ensure soundness.

6.3 Multiplication

Multiplication in fixed-point arithmetic is composed of two steps: 1) product computation of the fixed-point operands over integers, which results in a product with double the initial scale, followed by 2) a truncate-round step that restores the scale of the result. Formally, given two fixed-point integers x and y with scale s , $\gamma \triangleq \lfloor (x \cdot y) \cdot 2^{-s} \rfloor$, where $\lfloor \cdot \rfloor$ denotes that the result can be approximated into an integer in either direction. For sound interval analysis, we need to maintain both approximations of the result after truncation. However, multiplication is tricky over intervals since the intervals can be purely positive, purely negative or can even span zero. To account for all cases, the fixed-point interval multiplication is

$$C = A \otimes B \triangleq [\lfloor \min\{\underline{a}\underline{b}, \underline{a}\bar{b}, \bar{a}\underline{b}, \bar{a}\bar{b}\} \cdot 2^{-s} \rfloor, \lceil \max\{\underline{a}\underline{b}, \underline{a}\bar{b}, \bar{a}\underline{b}, \bar{a}\bar{b}\} \cdot 2^{-s} \rceil] \quad (3)$$

The soundness of the above multiplication follows from similar reasoning for floating-point interval multiplication [41]. Since we compute a min/max over all possible products of the interval endpoints, we account for all possible cases that arise from signs of the interval endpoints. The definitions of floor and ceiling maintain the soundness of the computation even after the truncation. Note that due to the monotonicity of truncation by 2^s , it suffices to truncate *just once* after the max/min computation of the four products.

6.3.1 Inner-Products

Throughout the DNN abstract interpretation using zonotopes, we will compute several inner-products between vectors of coefficients, weights and inputs. Since we are dealing with real-valued parameters and model inputs, it is important to compute inner-products between *vectors of fixed-point intervals* that over-approximate these values. We denote a vector of n fixed-point intervals by $\mathbf{A} = [A_1, \dots, A_n]$. Then, the inner-product between two length- n vectors $\mathbf{A}$ and $\mathbf{B}$ of fixed-point intervals is defined as

$$C = \langle \mathbf{A}, \mathbf{B} \rangle \triangleq \left[\left[\left\{ \bigoplus_{i=1}^n A_i \odot B_i \right\} \cdot 2^{-s} \right], \left[\left\{ \bigoplus_{i=1}^n A_i \overline{\odot} B_i \right\} \cdot 2^{-s} \right] \right], \quad (4)$$

where, for intervals A and B , the operations $A \odot B$ and $A_i \overline{\odot} B_i$ denote computing the minimum and maximum of the set $\{\underline{a}\underline{b}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b}\}$. Observe that the inner-product performs truncations *lazily*, i.e., only once after summing all element-wise products. Thus, the inner-product requires a single truncation regardless of the vector length. This optimization is useful for ZKPs of interval vector inner-products where truncation is an expensive non-linear operation.

Lemma 2. *The inner-product operator $\langle, \rangle$ is sound.*

6.4 Division

Since fixed-point numbers efficiently support integral multiplication, we model fixed-point division as a product between the fixed-point dividend and fixed-point *reciprocal* of the divisor. However, for DNN abstract interpretation with zonotopes, it suffices to deal with interval dividends and divisors that are purely positive. This assumption is easily relaxed to give a more generic fixed-point interval division over arbitrarily distributed intervals⁵. Given the positive real number b whose reciprocal is over-approximated by the fixed-point interval $B^{-1} = [\underline{\beta}, \overline{\beta}]$, interval division $A \oslash B$ is

$$C = A \oslash B \triangleq \left[\left[(\underline{a} \cdot \underline{\beta}) \cdot 2^{-s} \right], \left[(\overline{a} \cdot \overline{\beta}) \cdot 2^{-s} \right] \right] \quad (5)$$

Lemma 3. *The operator $\oslash$ is sound for purely positive intervals.*

Fixed-Point Reciprocal. We now define the reciprocal fixed-point interval for $x \in \mathbb{R}^*$. Formally, given $X = [\underline{x}, \overline{x}]$ over-approximating the positive real value x , the fixed-point reciprocal interval is defined as $X^{-1} \triangleq \left[\left\lfloor \frac{2^s}{x} \right\rfloor, \left\lceil \frac{2^s}{x} \right\rceil \right]$. Soundness of the reciprocal interval follows from definitions of floor and ceil.

Our sound reciprocal approximation is based on the lookup-table (LUT) evaluation approach of Hao et al. [25] which in turn draws from the widely used Newton-Raphson method [28, 36]

⁵However, for division to be defined over intervals, it is necessary that the divisor interval does not span zero.

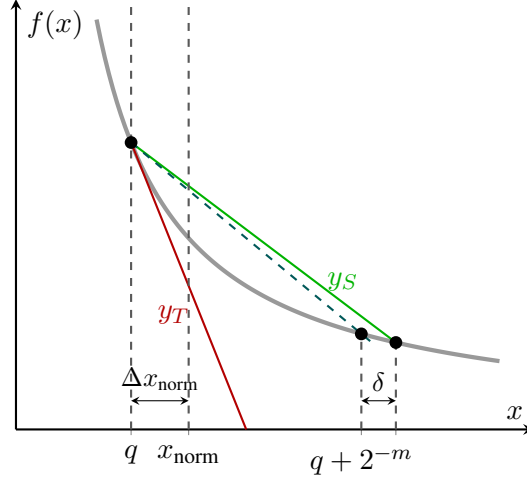


Figure 1. Visualization of our tangent y_T and secant y_S approximators for reciprocal with the shallow secant’s endpoint shifted by δ

and recent advancements in ZK-RAM [57]. We briefly review reciprocal computation using LUT evaluation here. For a positive real-value x , define its normalized form as $x_{\text{norm}} = x \cdot 2^{-k}$ such that $x_{\text{norm}} \in [1, 2)$. The interval $[1, 2)$ is divided into 2^m buckets of width 2^{-m} for $m \in \mathbb{Z}^+$. Each bucket is defined by an interval $[q, q + 2^{-m})$ where $q = 1 + \frac{i}{2^m}$ for $i \in \{0, \dots, 2^m - 1\}$. Then, x_{norm} is mapped to the bucket for which $x_{\text{norm}} \in [q, q + 2^{-m})$. $1/x_{\text{norm}}$ is approximated in this bucket using a linear approximation $y_R = a - b \cdot (x_{\text{norm}} - q)$. Essentially, the line y_r is defined in the point-slope form $y - \frac{1}{q} = m(x - q)$, where the slope m decides how closely the line straddles the curve $1/x$ in the interval $[q, q + 2^{-m})$. Thus, for each bucket $[q, q + 2^{-m})$, a line $y_R = a - b \cdot (x_{\text{norm}} - q)$ is defined whose coefficients (a, b) are stored in an LUT of size 2^m .

During the LUT evaluation, x_{norm} is parsed as $1 + r \cdot 2^{-m} + \Delta x_{\text{norm}}$ where $r \in \{0, \dots, 2^m - 1\}$ and $\Delta x_{\text{norm}} = x_{\text{norm}} - (1 + r \cdot 2^{-m})$ decides the distance from the start of the interval at which the linear approximation should be computed. We then fetch the coefficients (a, b) for the line y_R from the r -th row of the LUT, thus reconstructing the line straddling $1/x$ in the bucket $[1 + r \cdot 2^{-m}, 1 + r \cdot 2^{-m} + 2^{-m})$. By evaluating the linear expression $a - b \cdot \Delta x_{\text{norm}}$, we thus obtain the approximation of $1/x_{\text{norm}}$ given by y_R .

For a fixed-point number $\tilde{x} \in [2^k, 2^{k+1})$ of scale s , Hao et al. [25] evaluate the LUT in ZK by parsing the register representing the normalized number $z = \tilde{x} \cdot 2^{n-k-1} \in [2^{n-1}, 2^n)$ as $z = 1 \parallel z_2 \parallel z_1$ where $z_2 \in \{0, 1\}^m$ and $z_1 \in \{0, 1\}^{n-m-1}$. Their protocol then fetches the coefficients (a, b) from row z_2 of the LUT and computes $y = a - b \cdot z_1$. The approximation is denormalized by scaling the computed approximation of $1/x_{\text{norm}}$ with 2^{-k} to obtain the approximated value of $1/x$.

However, [25] does not perform sound interval analysis since their protocols are designed for ML inference. Thus, it suffices for them to compute a reasonably close approximation of $1/x$ using a line that “cuts” through the curve as closely as possible. This strategy does not work for us since the direction of approximation needs to be fixed for sound over-approximation. Instead, we specifically use the secant and tangent to the curve $1/x$ over the interval $[q, q + 2^{-m})$ to compute linear approximations of $1/x_{\text{norm}}$.

Algorithm 1 Tangent Reciprocal Approximation

- 1: **Parameters:** fixed-point scale s , fixed-point input bitlength n , LUT index bitlength m
 - 2: **Input:** $x \in (0, 2^{n-s})$
 - 3: **Start with the upper-bounding fixed-point integer**
 - 4: $\bar{x} = \lceil x \cdot 2^s \rceil$
 - 5: Find $k \in [0, n-1]$ such that $2^k \leq \bar{x} \leq 2^{k+1} - 1$
 - 6: $z \leftarrow \bar{x} \cdot 2^{n-1-k}$
 - 7: **LUT indexing**
 - 8: Parse $z = 1 \parallel z_2 \parallel z_1$, where $z_1 \in \{0, 1\}^{n-1-m}$ and $z_2 \in \{0, 1\}^m$
 - 9: $q \leftarrow 1 + \frac{z_2}{2^m}$
 - 10: $a \leftarrow \left\lfloor \frac{2^{n+s-1}}{q} \right\rfloor$, $b \leftarrow \left\lceil \frac{2^s}{q^2} \right\rceil$
 - 11: **Output scaling and denormalization**
 - 12: $y' \leftarrow (a - b \cdot z_1)$
 - 13: $y' \leftarrow \lfloor y' \cdot 2^{-(n-1)} \rfloor$
 - 14: $y' \leftarrow y' \cdot 2^{n-k}$
 - 15: **return** $\underline{y} \leftarrow \lfloor y' \cdot 2^{-(n-s)} \rfloor$
-

We now specify the computation of coefficients for the approximation lines in our division algorithm. For the underestimated reciprocal in the interval $[q, q + 2^{-m})$, we start with the upper-bounding divisor endpoint $\bar{x}$ and construct a tangent y_T to $1/x$ at $x = q$. This gives the coefficients in the point-slope form above as $a_0 = \frac{1}{q}$ (the value of $1/x$ at $x = q$ is $1/q$) and $a_1 = -\frac{\partial(1/x)}{\partial x}|_{x=q} = \frac{1}{q^2}$. Since the tangent to a convex curve (specifically $1/x$ in this case) is a global underestimator, the value of the reciprocal approximated from y_T is always less than the true reciprocal in the interval $[q, q + 2^{-m})$. For the overestimation, we start with the lower-bounding divisor endpoint $\underline{x}$ to construct a secant of $1/x$ between $x_0 = q$ and $x_1 = q + 2^{-m} + 2^{-k}$. The extra $\delta = 2^{-k}$ term is added to account for the least significant bits errors due to normalizing the *fixed-point* divisor $\underline{x}$. Since in such a case the real value of x_{norm} can exceed the value represented by the register z , the secant between $[q, q + 2^{-m}]$ can go slightly below $1/x$ for certain values of $\underline{x}$. Thus we construct a shallower secant where the right-endpoint is shifted to account for this truncation error. Thus, this secant $y_S = \frac{1}{q} - \frac{(x_{\text{norm}} - q)}{q(q + \epsilon)}$ always overestimates $1/x$ for all values of $\underline{x}$, where $\epsilon = 2^{-m} + 2^{-k}$.

While we lose some numerical precision due to this approximation, this strategy enables efficient reciprocal computation by leveraging recent advances in RAM-based zero-knowledge LUT protocols [25, 57]. We find empirically that the certification accuracy of our fixed-point zonotope abstraction remains comparable with that of the floating-point version which is far more precise (see Section 8). Algorithm 1 and Algorithm 2 detail our method for underestimation and overestimation of the reciprocal.

Lemma 4. *Algorithm 1 underestimates the real reciprocal for $x \in \mathbb{R}^+$, i.e., $\underline{y} \leq \frac{2^s}{x}$. Also, Algorithm 2 overestimates the real reciprocal for $x \in \mathbb{R}^+$, i.e., $\bar{y} \geq \frac{2^s}{x}$.*

Proofs of lemmas 1, 2 and 3 are deferred to Appendix D.

Algorithm 2 Secant Reciprocal Approximation

- 1: **Parameters:** fixed-point scale s , fixed-point input bitlength n , LUT index bitlength m
 - 2: **Input:** $x \in [2^{-s}, 2^{n-s})$
 - 3: **Start with the lower-bounding fixed-point integer**
 - 4: $\underline{x} = \lfloor x \cdot 2^s \rfloor$
 - 5: Find $k \in [0, n-1]$ such that $2^k \leq \underline{x} \leq 2^{k+1} - 1$
 - 6: $z \leftarrow \underline{x} \cdot 2^{n-1-k}$
 - 7: **LUT indexing**
 - 8: Parse $z = 1 \parallel z_2 \parallel z_1$, where $z_1 \in \{0, 1\}^{n-1-m}$ and $z_2 \in \{0, 1\}^m$
 - 9: $q \leftarrow 1 + \frac{z_2}{2^m}$
 - 10: $a \leftarrow \left\lceil \frac{2^{n+s-1}}{q} \right\rceil$, $b \leftarrow \left\lfloor \frac{2^s}{q(q+\epsilon)} \right\rfloor$ where $\epsilon = 2^{-m} + 2^{-k}$
 - 11: **Output scaling and denormalization**
 - 12: $y' \leftarrow (a - b \cdot z_1)$
 - 13: $y' \leftarrow \lceil (a - b \cdot z_1) \cdot 2^{-(n-1)} \rceil$
 - 14: $y' \leftarrow y' \cdot 2^{n-k}$
 - 15: **return** $\bar{y} \leftarrow \lceil y' \cdot 2^{-(n-s)} \rceil$
-

6.5 Interval Analysis over Fields

Our system proves abstract interpretation of neural networks in zero-knowledge. Since ZKPs typically operate over finite fields, we need to perform interval analysis over field elements for sound abstract interpretation. Interval analysis over field elements follows from the above specifications for fixed-point interval analysis using the mapping from fixed-point integers to field elements from Section 2.3. Similarly, the primitive operations such as addition and multiplication over integers are replaced by addition and multiplication over the field (typically performed as integer addition and multiplication modulo the prime p for the field $\mathbb{F}_p$). Non-linear operations such as truncation and division are performed in our system using similar techniques from Hao et al. [25] based on LUT evaluations (which are themselves computed using primitive operations of addition and multiplication over the field).

Remark. To prevent unsoundness due to wrap-arounds in the finite field, we use a large enough field modulus and select our fixed-point scale accordingly. This strategy is common in privacy-preserving ML literature [53]. Note, the unsoundness due to rounding errors discussed in Section 5 is an independent issue.

7 Zero-Knowledge Zonotope Abstraction

We first present the zonotope abstract domain [47] and the neural network specification. We then consider each abstract transformer separately alongside its translation to an interactive IT-MAC based ZKP. We compose these abstract transformers to present our zero-knowledge abstract analysis system AZKABAN.

7.1 The Zonotope Abstract Domain

A zonotope $\mathcal{Z} \in \mathbb{R}^n$ abstracts a set of n variables $\mathcal{X} \subseteq \mathbb{R}^n$ with an affine expression for each variable $x \in \mathcal{X}$ given by

$$x^{\mathcal{Z}} \triangleq \alpha_0 + \sum_{k=1}^t \alpha_k \cdot \epsilon_k, \text{ where } \forall k \in [t], \epsilon_k \in [-1, 1]. \quad (6)$$

Here, α_0 represents a center coefficient and $\{\epsilon_k\}_{k \in [t]}$ represent noise symbols in t independent directions that can vary between -1 and 1. Further, $\{\alpha_k\}_{k \in [t]}$ represent the noise coefficient by which the perturbation of the noise symbol ϵ_k is scaled. By sharing noise symbols between the affine forms of different variables, the zonotope models implicit inter-dependencies between the variables in $\mathcal{X}$. The affine form of variable x is concretized as follows to obtain the interval that over-approximates the set of x 's concrete values:

$$[\ell_x, u_x] \triangleq \left[\alpha_0 - \sum_{k=1}^t |\alpha_k|, \alpha_0 + \sum_{k=1}^t |\alpha_k| \right] \quad (7)$$

For DNN abstract interpretation, the zonotope domain abstracts each variable of the network, i.e., inputs, affine pre-activations, non-linear activations and output logits, using an affine expression as in Equation 6. DeepZono [47] preserves soundness w.r.t. floating-point numerical representation by over-approximating each real value above with floating-point intervals. Similarly, our ZK abstract interpreter operates over field intervals defined in Section 6.5.

7.2 Neural Network Specification

Let M be a DNN with d input features, h hidden layers with at most m neurons in each hidden layer and o output classes. $M : \mathbb{F}_p^d \rightarrow \mathbb{F}_p^o$ maps a d -dimensional vector of input features to an o -dimensional vector of logits, each of which quantifies the confidence of the DNN in the class. For an input $\mathbf{x}$, the classification/prediction $C(\mathbf{x})$ of the DNN is computed as $C(\mathbf{x}) = \underset{k \in [o]}{\operatorname{argmax}} \{M(\mathbf{x})_k\}$.

Each hidden neuron consists of two sequential computations. First, an *affine pre-activation* $v = \langle \mathbf{w}, \mathbf{x} \rangle + b$ where $\mathbf{w}, b, \mathbf{x}$ represent the neuron's weights, bias and previous layer's outputs respectively. Second, a *non-linear activation* $\sigma(v)$ computed on the output v of the affine pre-activation. Following DeepZono's convention, each neuron's affine and activation component is treated separately. Specifically, we expand the h -layer neural network into $(1 + 2h + 1)$ layers ordered as input $\rightarrow$ affine⁽¹⁾ $\rightarrow$ activation⁽¹⁾ $\rightarrow \dots \rightarrow$ affine^(h) $\rightarrow$ activation^(h) $\rightarrow$ output, where the layer index is written in superscript. We use the term 'variable' to uniformly refer to an input, affine pre-activation, activation or output computation.

Notations. In our analysis, each variable v_i is assigned an index $i \in \mathbb{Z}^+$ in a topological order starting from the index 1 assigned to the first input variable. The analysis progresses layer-by-layer, assigning an index (the next largest integer) to each variable in one layer before moving to the next layer. An index set $I \subset \mathbb{Z}^+$ represents the set of variables v_i such that $i \in I$. Throughout our analysis, we maintain a set of noise symbol indices that keeps growing as the analysis progresses in the topological order above. Formally, let $\mathcal{N}_i \subset \mathbb{Z}^+$ contain the indices of the noise symbols in the affine expression for the variable v_i . When abstracting the i -th variable in the DNN (i.e., i is the largest assigned index), the set $\mathcal{N} = \bigcup_{j=1}^{i-1} \mathcal{N}_j$ contains the indices of all noise symbols present in the affine expressions of the variables that have been abstracted.

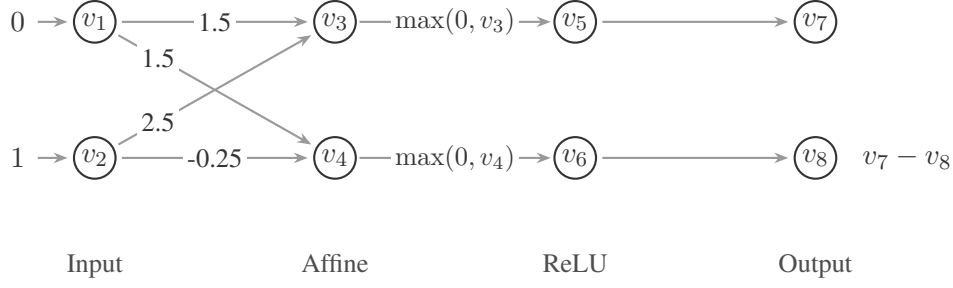


Figure 2. A toy neural network.

7.3 Input Layer

DeepZono certifies local adversarial robustness in an ℓ_∞ -box of radius $\delta \in \mathbb{R}$. Thus, for each input variable v_i , $i \in [d]$, the abstract input element is constructed by over-approximating the set of input perturbations as $[x_i - \delta, x_i + \delta]$. Therefore, the set of affine expressions for the input layer's variables is defined as $\{v_i^{\mathcal{Z}} \triangleq x_i + \delta \cdot \epsilon_i\}_{i \in [d]}$, where the i -th input feature $\{x_i \in \mathbb{R}\}_{i \in [d]}$ forms the center coefficient α_{i0} for the variable v_i . Similarly, δ is the noise coefficient α_{i1} for ϵ_i . For ensuring soundness, we use field intervals to over-approximate the coefficients as follows:

$$\{V_i^{\mathcal{Z}} \triangleq X_i \oplus \partial \odot \epsilon_i\}_{i \in [d]}, \quad (8)$$

where $\{X_i = [x_i, \bar{x}_i] \subset \mathbb{F}_p\}_{i \in [d]}$ and $\partial = [\underline{\delta}, \bar{\delta}] \subset \mathbb{F}_p$. Since both the input vector and perturbation radius are public constants, $\mathcal{P}$ and $\mathcal{V}$ obtain IT-MACs for X_i and ∂ non-interactively for free.

Running Example (1). Consider the real-valued toy DNN in Figure 2 and scale $s = 4$ for fixed-point arithmetic. The perturbation radius is $\Delta = [4, 4]$ and the input variables v_1 and v_2 are abstracted with the affine expressions: $V_1^{\mathcal{Z}} = [0, 0] \oplus [4, 4] \odot \epsilon_1$ and $V_2^{\mathcal{Z}} = [16, 16] \oplus [4, 4] \odot \epsilon_2$ (as per Equation 8).

Authenticating Parameter Intervals. $\mathcal{P}$ authenticates all model parameters in terms of field intervals before $\mathcal{V}$ issues his query. Observe that $\mathcal{P}$ can cheat by committing to intervals $[\underline{w}, \bar{w}]$ that are not well-formed, i.e., $\underline{w} > \bar{w}$. We prevent this by making $\mathcal{P}$ and $\mathcal{V}$ validate for all parameters $w \in \mathbb{R}$ that the $1\{\bar{w} \geq \underline{w}\}$ holds over authenticated $\underline{w}, \bar{w}$.

7.4 Affine Pre-Activation

The affine pre-activation variable v_i (for a neuron with weights $\mathbf{w}_i$ and bias b_i) is computed as $\langle \mathbf{w}_i, \mathbf{x}' \rangle + b_i$, where $\mathbf{x}'$ represents the previous layer's output vector. In the Zonotope domain, the previous layer's variables $\{v_j\}_{j \in I}$ (for an index set I) are abstracted by their respective affine expressions $\{v_j^{\mathcal{Z}}\}_{j \in I}$. Let $\mathcal{N}^{\text{prev}} = \bigcup_{j \in I} \mathcal{N}_j$. Then, the affine expression for a pre-activation variable v_i is defined as (for brevity, we directly present the expression over field intervals):

$$V_i^{\mathcal{Z}} \triangleq \underbrace{(B_i \oplus \langle \mathbf{W}_i, \mathbf{A}_0 \rangle)}_{A_{i0}} \oplus \sum_{k \in \mathcal{N}^{\text{prev}}} \underbrace{\langle \mathbf{W}_i, \mathbf{A}_k \rangle}_{A_{ik}} \odot \epsilon_k. \quad (9)$$

Here, $\mathbf{A}_k = (A_{jk})_{j \in I}$ denotes the vector of intervals that over-approximate the coefficient of noise symbol ϵ_k in the affine expression of each variable in the previous layer. Similarly, $\mathbf{A}_0 =$

$(A_{j0})_{j \in I}$ is the vector of center-coefficient intervals from the previous layer. $B_i = [b_i, \overline{b_i}]$ and $\forall j \in I$, $W_{ij} = [w_{ij}, \overline{w_{ij}}]$ over-approximate the bias and weights of the corresponding neuron. Since the pre-activation computation simply computes a linear combination of the variables, and consequently, their centers and coefficients, it does not introduce any new noise symbols. Thus, the affine expression for v_i consists of the center coefficient A_{i0} and noise coefficients $\{A_{i1}, \dots, A_{i|\mathcal{N}^{\text{prev}}|}\}$. We later show in Section 7.5 that non-linear activations can in fact introduce new noise symbols.

Running Example (2). The weights for v_3 and v_4 are $\mathbf{W}_3 = ([24, 24], [40, 40])$ and $\mathbf{W}_4 = ([24, 24], [-4, -4])$. Thus, applying Equation 9, the affine expression for v_3 is $V_3^{\mathcal{Z}} = [24, 24] \otimes V_1^{\mathcal{Z}} + [40, 40] \otimes V_2^{\mathcal{Z}}$. Substituting $V_1^{\mathcal{Z}}$ and $V_2^{\mathcal{Z}}$ from **Running Example (1)** gives us $V_3^{\mathcal{Z}} = [24, 24] \otimes \{[0, 0] \oplus [4, 4] \odot \epsilon_1\} \oplus [40, 40] \otimes \{[16, 16] \oplus [4, 4] \odot \epsilon_2\} = [40, 40] + [6, 6] \odot \epsilon_1 + [10, 10] \odot \epsilon_2$. Similarly, $V_4^{\mathcal{Z}} = [-4, -4] \oplus [6, 6] \odot \epsilon_1 \oplus [-1, -1] \oplus \epsilon_2$.

Interval Multiplication in Zero-Knowledge. Recall that interval multiplication involves computing a minimum and maximum of the four combinations of end-point products. In our system, we first compute these four products $\tau_1, \tau_2, \tau_3, \tau_4 \in \mathbb{F}_p^4$ and perform a consolidated minmax computation over them to reuse comparison results (see Protocol 4). The zero-knowledge proof for the *minmax* computation thus uses 4 comparisons (we use the LUT-based protocol of [25]) and 4 field multiplications (additions are free due to homomorphism of IT-MACs). Thus, one interval multiplication requires 8 field multiplications and 4 comparisons.

Truncation in Zero-Knowledge. Recall that fixed-point interval multiplication requires truncation with both rounding up and rounding down for soundness. Hao et al. [25] give an LUT-based ZKP for truncation with rounding down. We augment their protocol to compute truncation with rounding up as follows. In particular, given $t \in \mathbb{Z}$, let $\underline{t}$ denote the integer $\lfloor -t \cdot 2^{-s} \rfloor$. Then, using the identity over reals $\lceil x \rceil = -\lfloor -x \rfloor$, we obtain $\bar{t} = \lceil t \cdot 2^{-s} \rceil = -\lfloor -t \cdot 2^{-s} \rfloor = -\underline{t}$. We then simply translate this result to computation over $\mathbb{F}_p$ using the mapping between fixed-point integers and field elements (Section 2.3). Since negating a field element in $\mathbb{F}_p$ (multiplication by the scalar $p - 1$) is free in IT-MAC based ZKPs, we enable sound interval multiplications by black-box use of [25]’s truncation protocol that only realizes truncation with rounding-down.

Interval Inner-Products in Zero-Knowledge. Recent works in VOLE-based ZKPs have reduced the cost of inner-products of field element vectors (i.e., $\mathbb{F}_p^n$) to the communication of single IT-MAC in addition to committing to the witness vectors [56]. However, proving inner-products between vectors of field *intervals* (i.e., $(\mathbb{F}_p^2)^n$) while maintaining both soundness and precision add significant proof costs. This cost stems from the requirement of computing element-wise product combinations followed by a non-linear min/max computation as shown in Section 6.3. Since all intervals in the vectors are independent of each other with each interval in a different state based on its endpoint signs, it is not sound to vectorize the 4 product computations in 4 (which would have been extremely efficient due to [56]), and then compute a min/max over the 4 inner-products. This problem is similar to that of MLPs where a sequence of affine transformations interleaved with non-linear activations cannot be collapsed into a single affine transformation. There exist other interval representations such as the midpoint-radius form [43] that also do sound interval arithmetic with concretely lesser number of comparisons per vector coordinate; however, they are too imprecise to be useful in DNN analysis. We leave open the problem of finding concrete improvements for ZK interval inner-products for future work.

7.5 ReLU Activation

The ReLU activation function is defined as $\text{ReLU}(x) = \max(0, x)$. Let the variable v_i represent the ReLU activation of the previous (affine) variable v_j , i.e., $v_i = \text{ReLU}(v_j)$ where $j < i$. Let $[\ell_j, u_j]$ denote the concretization of the affine expression for v_j from equation 7. Then, DeepZono specifies the affine expression for v_i as

$$v_i^{\mathcal{Z}} \triangleq \begin{cases} v_j^{\mathcal{Z}} & \text{if } \ell_j \geq 0 \\ 0 & \text{if } u_j \leq 0 \\ \lambda \cdot v_j^{\mathcal{Z}} + \gamma + \gamma \cdot \epsilon_{t+1} & \text{if } \ell_j < 0 < u_j \end{cases}$$

where $\lambda \triangleq \frac{u_j}{\ell_j - u_j}$, $\gamma \triangleq \frac{-u_j \ell_j}{2(u_j - \ell_j)}$ and $t = |\mathcal{N}|$. In the third case where v_j can be either positive or negative, DeepZono approximates the ReLU variable by introducing a new noise symbol ϵ_{t+1} .

Neuron State Leakage. Since the number of resulting noise symbols is different in each case, the number of field elements required to authenticate the affine expression for v_i implicitly leaks the state of v_j . This leakage is significant since previous works have shown extraction of DNN parameters by exploiting the neural states [9, 12], thus defeating our model privacy goal. We address this leakage by padding zero-coefficient noise symbols to the affine expressions in the other two cases. In particular, given $\mathcal{N}^{\text{prev}}$ as the set of v_j 's and the affine expression $V_j^{\mathcal{Z}} \triangleq A_{j0} \oplus \sum_{k \in \mathcal{N}^{\text{prev}}} A_{jk} \odot \epsilon_k$ for variable v_j ,

$$V_i^{\mathcal{Z}} \triangleq \begin{cases} A_{j0} \oplus (\sum_{k \in \mathcal{N}^{\text{prev}}} A_{jk} \odot \epsilon_k) \oplus [0, 0] \odot \epsilon_{t+1} & \text{if } \ell_j \geq 0 \\ [0, 0] \oplus (\sum_{k \in \mathcal{N}^{\text{prev}}} [0, 0] \odot \epsilon_k) \oplus [0, 0] \odot \epsilon_{t+1} & \text{if } u_j \leq 0 \\ (\Lambda \otimes A_{j0} + \Gamma) \oplus (\sum_{k \in \mathcal{N}^{\text{prev}}} \odot \epsilon_k) \oplus (\Gamma \odot \epsilon_{t+1}) & \text{if } \ell_j < 0 < u_j \end{cases}$$

where $\Lambda = [\underline{\lambda}, \bar{\lambda}]$ and $\Gamma = [\underline{\gamma}, \bar{\gamma}]$. Due to the hiding property of IT-MACs, $\mathcal{V}$ cannot distinguish a non-zero authenticated coefficient from a zero-coefficient. Since all possible affine expressions of a ReLU variable now have equal number of coefficients, the three cases above look indistinguishable to $\mathcal{V}$, thus preventing leakage. Details on ZK concretization (Alg. 3) are deferred to Appendix E.

Running Example (3). For the ReLU variables v_5 and v_6 we first compute concrete bounds of $[\ell_3, u_3] = [24, 56]$ and $[\ell_4, u_4] = [-11, 3]$ using equation 7 over fixed-point intervals. Since the first interval is purely positive, we get $V_5^{\mathcal{Z}} = V_3^{\mathcal{Z}} = [40, 40] \oplus [6, 6] \odot \epsilon_1 \oplus [10, 10] \odot \epsilon_2$. However, since $[-11, 3]$ spans zero, the affine expression $V_6^{\mathcal{Z}}$ adds a new noise symbol and coefficients that are computed using interval division. For brevity, we just show the standard fixed-point division over intervals here, i.e., $\lfloor (A \otimes [2^s, 2^s]) / B \rfloor$ and skip our LUT-based division. In particular, we compute coefficients (corresponding to the slope λ) $\Lambda = ([u_4, u_4] \otimes [16, 16]) \odot ([u_4, u_4] \ominus [\ell_4, \ell_4]) = [3, 4]$. Similarly, we get the interval $\Gamma = [1, 2]$ corresponding to the intercept γ . Thus, we obtain the zonotope for the ReLU variable v_6 : $V_6^{\mathcal{Z}} = [3, 4] \otimes V_4^{\mathcal{Z}} \oplus \Gamma \oplus \Gamma \odot \epsilon_3 = [0, 3] \oplus [1, 2] \odot \epsilon_1 \oplus [-1, 1] \odot \epsilon_2 \oplus [1, 2] \odot \epsilon_3$, where ϵ_3 is a new noise symbol and the noise symbols are not concretized yet. Thus, to ensure zero-knowledge we pad $V_5^{\mathcal{Z}}$ to $V_5^{\mathcal{Z}} = [40, 40] \oplus [6, 6] \odot \epsilon_1 \oplus [10, 10] \odot \epsilon_2 \oplus [0, 0] \odot \epsilon_3$.

We leave extensions to other activation functions as future work.

7.6 Output

The input $\mathbf{x}$ is certified for the property ψ , i.e., for robustness in radius δ , in the output layer. For a DNN with C output classes, the safe output set consists of all outputs with the same label $\hat{y}$ as the

input x , i.e., $S_\psi = \{\mathbf{y} \in \mathbb{R}^C \mid \arg \max_{i \in [C]} (y_i) = \hat{y}\}$, where $\mathbf{y}$ denotes the output logits of the DNN. Essentially, an input $\mathbf{x}$ is certified safe if for all adversarial inputs in the radius δ around $\mathbf{x}$, the DNN produces the same output prediction.

The output variables simply copy the corresponding variables in the previous layer. In DeepZono, the output layer constructs a set of difference variables which abstract the difference of the j -th output variable $v_{\text{out}j}$ ($j \in [o]$) from the ground truth output variable v_g . Since $v_g - v_{\text{out}j}$ is an affine transformation on two variables, the corresponding ZKP follows from Section 7.4. If the concretization $[\ell_{\text{diff}j}, u_{\text{diff}j}]$ of class j 's difference variable is purely positive, by soundness of overapproximation $y_g - y_{\text{out}j} > 0$ holds for all input perturbations. The input is certified safe if this holds for all the difference variables, i.e., the conjunction $\bigwedge_{j \in [o]} \{1\{\ell_j > 0\}\}$.

Running Example (4). We assume the ground truth to be 0 i.e., the first output logit must be the largest. The difference variables in the output layer are abstracted as $V_{\text{diff}0}^Z = V_7^Z \ominus V_7^Z = [0, 0] \oplus [0, 0] \odot \epsilon_1 \oplus [0, 0] \odot \epsilon_2 \oplus [0, 0] \odot \epsilon_3$ and $V_{\text{diff}1}^Z = V_7^Z \ominus V_8^Z = [37, 40] \oplus [4, 5] \odot \epsilon_1 \oplus [9, 11] \odot \epsilon_2 \oplus [-2, -1] \odot \epsilon_3$. Concretizing both expressions, we get $[\ell_{\text{diff}0}, u_{\text{diff}0}] = [0, 0]$ and $[\ell_{\text{diff}1}, u_{\text{diff}1}] = [19, 57]$ which are both purely positive. Thus, the input is certified safe.

7.7 Putting Everything Together

Composing the above abstract transformers completes the abstract interpretation of the DNN on the given input. Our system models the operations above as an arithmetic circuit over $\mathbb{F}_p$ which operates over IT-MAC commitments of the parameters, inputs and intermediate results. Protocol 5 formalizes our ZKP for abstract interpretation in the zonotope domain.

Theorem 5. *Protocol 5 is a zero-knowledge proof system for DNN abstract interpretation in the zonotope abstract domain.*

Proof. (Sketch). Our system for ZK abstract interpretation constructs a circuit over $\mathbb{F}_p$ that emulates the interval operators from Section 6 to compose abstract transformers of the different DNN layers in Section 7. We then evaluate the circuit using existing VOLE-based ZK proof systems [15, 25, 56]. Completeness, soundness and zero-knowledge properties of our system follow:

Completeness follows from the completeness of underlying ZKP. While abstract interpretation itself is incomplete, inputs that can be certified in the zonotope domain can be certified in zero-knowledge.

Soundness follows from two observations. First, our constructed circuit achieves soundness of abstract interpretation w.r.t. reals. This follows from the soundness of the abstract transformers in DeepZono and the soundness of interval addition, inner-products and division over fixed-point intervals from lemmas 1, 2, 3, 4. Second, soundness of the proof system for circuits over $\mathbb{F}_p$ follows from soundness of the underlying ZKP. Thus, the overall ZKP for abstract interpretation satisfies soundness w.r.t. reals.

Similarly, *zero-knowledge* follows from two observations. First, the constructed circuit is independent of the secret witness (i.e., model parameters) and thus does not leak information about the witness. Second, we simply apply a general VOLE ZK protocol to prove execution of the above circuit. $\square$

8 Evaluation

8.1 Implementation and Setup

AZKABAN is implemented in C++ on top of the EMP toolkit [1]. We heavily leverage the protocol implementations included therein alongside the open-source implementation of [25] for functionalities such as comparison, exponentiation and truncation. Since our framework is based on the DeepZono [47] algorithm, we use its implementation from the ERAN codebase [2] for benchmarking certification accuracy. We use scale $s = 24$ for fixed-point math. For division, we use $n = 30$ and $m = 11$ for indexing the LUT.

We run our tests on an AMD machine equipped with an EPYC 8434P processor and running Ubuntu 24.04.3 LTS. The prover and verifier communicate over a LAN network with 1 Gbps bandwidth simulated using Linux’s `tc` command. Our code is single-threaded and does not use any parallelization.⁶ We present results of evaluating our framework for robustness followed by fairness.

8.2 Adversarial Robustness

Datasets and Models. Similar to ERAN, we evaluate our framework for certifying adversarial robustness using adversarial perturbations of images from the MNIST [39] and CIFAR-10 [38] datasets. We evaluate adversarial robustness of 6 different neural network architectures which we list in Table 2. Models R1-R4 are multi-layer perceptrons (MLPs) while models R5-R6 are convolutional neural networks (CNNs). We remark that models R4 and R5 are trained by us using PyTorch for 10 epochs using Adam optimizer and a learning rate of 0.001 and batch-size 128. All other models in Table 2 are taken from ERAN. The perturbation radius δ is chosen to achieve useful certification accuracy computed as:

$$\text{Certification Accuracy (\%)} = \frac{\# \text{ inputs certified safe} \times 100}{\# \text{ inputs correctly classified}} \quad (10)$$

Robustness results. We present in Table 2 the certification accuracy and proof costs of our framework. The test datasets for MNIST and CIFAR-10 are taken from ERAN which consist of 100 images from each dataset. Since there does not exist a baseline for ZK robustness certification, we only benchmark our system’s certification accuracy against the cleartext baseline from ERAN. Further, we mention the proof costs in terms of end-to-end proof time and total communication between $\mathcal{P}$ and $\mathcal{V}$. These metrics are inclusive of costs for parameter authentication and setting up tables for LUT-based protocols. Table 2 shows that the zero-knowledge certification accuracy of AZKABAN matches ERAN’s, which is computed over double-precision floats. This is attributed to tight fixed-point interval analysis that closely parallels the precision of double-precision floating-point analysis due to the choice of the fixed-point scale.

8.3 Fairness

We evaluate AZKABAN for zero-knowledge certification of local individual fairness (LIF) introduced in FairProof [55]. LIF quantifies DNN fairness by measuring the minimum perturbation to the non-sensitive attributes of the input under which the model output remains constant. Since this

⁶While DeepZono is completely parallelizable, we leave this extension for AZKABAN to future work.

Table 2. Certification accuracy and proof costs for robustness.

Dataset	Model	Neur.	δ	Accuracy (%)		Proof Cost	
				ERAN	Ours	Time	Comm.(GB)
MNIST	R1	110	0.01	86	86	2 m	3.47
	R2	210	0.01	73	73	4 m	11.87
	R3	510	0.01	46	45	17 m	50.49
CIFAR-10	R4	410	0.0002	100	100	20 m	57.76
	R5	2312	0.002	90	90	2 hr	386.89
	R6	4852	0.002	96	96	4 hr	622.83

Table 3. Fairness certification and proof costs.

Dataset	Model	Neurons	Proof Time (sec)		
			FairProof	Ours	Improv.
Adult	F1	12	330	3	110×
	F2	112	—	6	—
Credit	F3	8	180	3	60×
	F4	112	—	8	—
German	F5	8	120	3	40×
	F6	112	—	7	—

measurement is reducible to local adversarial robustness (see [55] for the reduction), AZKABAN certifies LIF using DeepZono-based abstract interpretation.

Models and Datasets. For LIF, we consider the same datasets as FairProof: Adult [7], Credit Default [58] and German Credit [27]. These datasets consist of categorical and numerical attributes. We use the asme sensitive binary attributes “gender” for Adult and Credit Default datasets and “foreign worker” for the German Credit dataset as [55], and also use their reported fairness threshold. Models F1, F3 and F5 in Table 3 represent 2-layer model architectures with (8+2) and (4+2) neurons, as per FairProof’s evaluation [55]. Since their models and training code are not public, we train our own models with these architectures following their settings. However, we were unable to run FairProof on our models because the model loading script is not provided. We thus used their reported proof times for these models. FairProof reports on Intel i9 with a 3.60 GHz up to 5.00 GHz CPU clock speed; we run on EPYC CPU with max frequency 3.1GHz. Models F2, F4 and F6 contain 2 hidden layers and 50 neurons per layer. Since we were unable to run FairProof on these models, these are marked as “-” in Table 3.

Robustness results. Table 3 shows the proof costs of AZKABAN in comparison with the reported FairProof results. AZKABAN certifies LIF 40 – 110× faster than FairProof on the evaluated models (Table 3). Moreover, AZKABAN scales to a much larger neural network with 112 neurons and proves fairness certification in a few seconds, while maintaining end-to-end soundness over fixed-point arithmetic.

9 Related Work

FairProof. The state-of-the-art zero-knowledge individual fairness certification protocol FairProof [55] studies *local individual fairness* (LIF) that quantifies a model’s fairness towards bounded-distance perturbations around non-sensitive attributes of a given input (hence, local). FairProof is based on the GeoCert [33] local robustness certification algorithm which produces a perturbation radius under which the DNN is robust for the given input – i.e., it supports *complete* certification whereas AZKABAN (based on incomplete abstract interpretation) can fail to certify certain inputs. However, due to their SNARK-based design of proving a search problem in a polyhedral complex that is exponentially large in the neural count of the DNN, FairProof scales to only tiny models with 6 – 10 neurons. More importantly, FairProof (and GeoCert) is not sound w.r.t. fixed/floating-point numbers.

FairZK. Group fairness [26] metrics compare the DNN’s *aggregated* behavior between subgroups differing on a sensitive attribute. FairZK [59] is the state-of-the-art SNARK-based protocol for provable estimation of an upper-bound on the statistical parity (SP) [16] of a DNN. While FairZK scales well to million-parameter DNNs, the estimated SP bounds are too loose to be useful for absolute quantification of DNN fairness. Second, even FairZK does not account for the numerical imprecision introduced by fixed-point multiplications which accumulate over DNN successive layers. Since FairZK studies group fairness, our work doesn’t directly compare with it.

OATH. Finally, OATH [19] studies ZK certification of demographic parity which also quantifies group fairness. However, OATH is almost $4 - 5\times$ slower than SNARK-based FairZK.

Adversarial robustness. Another useful property of neural networks is adversarial robustness [11, 49] that measures the stability of a DNN’s output w.r.t. adversarially perturbed inputs. Several works have explored DNN robustness certification with full access to the DNN parameters [20, 33, 34, 47, 48, 51]. However, to our knowledge, there do not exist ZK robustness certification protocols.

10 Conclusion

In this work, we propose sound protocols for ML auditing that maintain model confidentiality. We identified soundness gaps in existing literature on zero-knowledge fairness certification protocols. We identified strategies to overclaim model fairness in two protocols FairProof and FairZK by exploiting rounding errors in fixed-point arithmetic. Highlighting the need for careful selection of rounding modes, we propose operators for fixed-point interval arithmetic that soundly overapproximate the real-valued computations, including ZKP-amenable algorithms for interval inner-products and LUT-based division. Using these interval operators, we construct AZKABAN— a system for zero-knowledge DNN abstract interpretation based on the DeepZono algorithm. We evaluate our system for certifying robustness and individual fairness of MLPs and CNNs. We improve over the state-of-the-art individual fairness certification protocol by $40 - 110\times$ in end-to-end proof time while maintaining certification soundness.

Acknowledgements

This work was supported in part by the National Science Foundation (NSF) Cybersecurity Innovation for Cyberinfrastructure program under Grant No. 2531010, by a Visa research award, and by NSF

awards CNS-2246354 and CCF-2217070. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

- [1] EMP-Toolkit. URL <https://github.com/emp-toolkit>.
- [2] ERAN: ETH robustness analyzer for neural networks. URL <https://github.com/eth-sri/eran>.
- [3] IEEE standard for floating-point arithmetic. *IEEE Std 754-2008*, pages 1–70, 2008. doi: 10.1109/IEEESTD.2008.4610935.
- [4] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias: There’s software used across the country to predict future criminals. and it’s biased against blacks. <https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing>, 2016. Accessed 2026-04-28.
- [5] Robert Bartlett, Adair Morse, Richard Stanton, and Nancy Wallace. Consumer-lending discrimination in the FinTech era. *Journal of Financial Economics*, 143(1):30–56, 2022. doi: 10.1016/j.jfineco.2021.05.047.
- [6] Osbert Bastani, Xin Zhang, and Armando Solar-Lezama. Probabilistic verification of fairness properties via concentration. *Proceedings of the ACM on Programming Languages*, 3 (OOPSLA):1–27, 2019.
- [7] Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: <https://doi.org/10.24432/C5XW20>.
- [8] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 169–188. Springer, 2011.
- [9] Isaac A Canales-Martínez, Jorge Chávez-Saab, Anna Hambitzer, Francisco Rodríguez-Henríquez, Nitin Satpute, and Adi Shamir. Polynomial time cryptanalytic extraction of neural network models. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–33. Springer, 2024.
- [10] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 494–503, 2002.
- [11] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. Ieee, 2017.
- [12] Nicholas Carlini, Matthew Jagielski, and Ilya Mironov. Cryptanalytic extraction of neural network models. In *Annual international cryptology conference*, pages 189–218. Springer, 2020.

- [13] Patrick Cousot and Radhia Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 238–252, 1977.
- [14] Jeffrey Dastin. Amazon scraps secret AI recruiting tool that showed bias against women. <https://www.reuters.com/article/world/insights/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK0AG/>, October 2018. Accessed 2026-04-28.
- [15] Samuel Dittmer, Yuval Ishai, and Rafail Ostrovsky. Line-point zero knowledge and its applications. *Cryptology ePrint Archive*, 2020.
- [16] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.
- [17] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *nature*, 542(7639):115–118, 2017.
- [18] Samuel G Finlayson, John D Bowers, Joichi Ito, Jonathan L Zittrain, Andrew L Beam, and Isaac S Kohane. Adversarial attacks on medical machine learning. *Science*, 363(6433):1287–1289, 2019.
- [19] Olive Franzese, Ali Shahin Shamsabadi, Carter Luck, and Hamed Haddadi. Secure and confidential certificates of online fairness. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL <https://openreview.net/forum?id=9jaQTx108T>.
- [20] Timon Gehr, Matthew Mirman, Dana Drachler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2018. doi: 10.1109/SP.2018.00058.
- [21] Khalil Ghorbal, Eric Goubault, and Sylvie Putot. The zonotope abstract domain taylor1+. In *International conference on computer aided verification*, pages 627–633. Springer, 2009.
- [22] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing, STOC '85*, page 291–304, New York, NY, USA, 1985. Association for Computing Machinery. ISBN 0897911512. doi: 10.1145/22145.22178. URL <https://doi.org/10.1145/22145.22178>.
- [23] Kanav Gupta, Deepak Kumaraswamy, Nishanth Chandran, and Divya Gupta. Llama: A low latency math library for secure inference. *Cryptology ePrint Archive*, 2022.
- [24] Kanav Gupta, Neha Jawalkar, Ananta Mukherjee, Nishanth Chandran, Divya Gupta, Ashish Panwar, and Rahul Sharma. Sigma: Secure gpt inference with function secret sharing. *Cryptology ePrint Archive*, 2023.

- [25] Meng Hao, Hanxiao Chen, Hongwei Li, Chenkai Weng, Yuan Zhang, Haomiao Yang, and Tianwei Zhang. Scalable zero-knowledge proofs for non-linear functions in machine learning. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 3819–3836, 2024.
- [26] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29, 2016.
- [27] Hans Hofmann. Statlog (German Credit Data). UCI Machine Learning Repository, 1994. DOI: <https://doi.org/10.24432/C5NC77>.
- [28] Masayuki Ito, Naofumi Takagi, and Shuzo Yajima. Efficient initial approximation for multiplicative division and square root by a multiplication with operand modification. *IEEE Transactions on Computers*, 46(4):495–498, 2002.
- [29] Neha Jawalkar, Kanav Gupta, Arkaprava Basu, Nishanth Chandran, Divya Gupta, and Rahul Sharma. Orca: Fss-based secure training and inference with gpus. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 597–616. IEEE, 2024.
- [30] Kai Jia and Martin Rinard. Exploiting verified neural networks via floating point numerical error. In *International Static Analysis Symposium*, pages 191–205. Springer, 2021.
- [31] Jiankai Jin, Eleanor McMurtry, Benjamin IP Rubinstein, and Olga Ohrimenko. Are we there yet? timing and floating-point attacks on differential privacy systems. In *2022 IEEE Symposium on security and privacy (SP)*, pages 473–488. IEEE, 2022.
- [32] Jiankai Jin, Olga Ohrimenko, and Benjamin IP Rubinstein. Getting a-round guarantees: floating-point attacks on certified robustness. In *Proceedings of the 2024 Workshop on Artificial Intelligence and Security*, pages 53–64, 2024.
- [33] Matt Jordan, Justin Lewis, and Alexandros G Dimakis. Provable certificates for adversarial examples: Fitting a ball in the union of polytopes. *Advances in neural information processing systems*, 32, 2019.
- [34] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International conference on computer aided verification*, pages 97–117. Springer, 2017.
- [35] Amir E. Khandani, Adlar J. Kim, and Andrew W. Lo. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34(11):2767–2787, 2010. doi: 10.1016/j.jbankfin.2010.06.001.
- [36] Israel Koren. *Computer arithmetic algorithms*. AK Peters/CRC Press, 2018.
- [37] O Deniz Kose and Yanning Shen. Fairgat: Fairness-aware graph attention networks. *ACM Transactions on Knowledge Discovery from Data*, 18(7):1–20, 2024.
- [38] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [39] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2002.

- [40] Tianyi Liu, Xiang Xie, and Yupeng Zhang. Zkcnn: Zero knowledge proofs for convolutional neural network predictions and accuracy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2968–2985, 2021.
- [41] Antoine Miné. Relational abstract domains for the detection of floating-point run-time errors. In *European Symposium on Programming*, pages 3–17. Springer, 2004.
- [42] Ilya Mironov. On significance of the least significant bits for differential privacy. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 650–661, 2012.
- [43] Ramon E Moore and CT Yang. Interval analysis i. *Technical Document LMSD-285875, Lockheed Missiles and Space Division, Sunnyvale, CA, USA*, 1959.
- [44] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In *Annual Cryptology Conference*, pages 681–700. Springer, 2012.
- [45] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, et al. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *arXiv preprint arXiv:1711.05225*, 2017.
- [46] Arman Riasi, Haodi Wang, Rouzbeh Behnia, Viet Vo, and Thang Hoang. Zero-knowledge ai inference with high precision. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pages 1053–1067, 2025.
- [47] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. *Advances in neural information processing systems*, 31, 2018.
- [48] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL): 1–30, 2019.
- [49] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.
- [50] Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.
- [51] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *Advances in neural information processing systems*, 34:29909–29921, 2021.
- [52] Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1074–1091. IEEE, 2021.

- [53] Chenkai Weng, Kang Yang, Xiang Xie, Jonathan Katz, and Xiao Wang. Mystique: Efficient conversions for $\{\text{Zero-Knowledge}\}$ proofs with applications to machine learning. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 501–518, 2021.
- [54] Haoze Wu, Omri Isac, Aleksandar Zeljić, Teruhiro Tagomori, Matthew Daggitt, Wen Kokke, Idan Refaeli, Guy Amir, Kyle Julian, Shahaf Bassan, et al. Marabou 2.0: a versatile formal analyzer of neural networks. In *International Conference on Computer Aided Verification*, pages 249–264. Springer, 2024.
- [55] Chhavi Yadav, Amrita Roy Chowdhury, Dan Boneh, and Kamalika Chaudhuri. FairProof : Confidential and certifiable fairness for neural networks. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 55682–55705. PMLR, 21–27 Jul 2024.
- [56] Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. Quicksilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2986–3001, 2021.
- [57] Yibin Yang and David Heath. Two shuffles make a $\{\text{RAM}\}$: Improved constant overhead zero knowledge $\{\text{RAM}\}$. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 1435–1452, 2024.
- [58] I-Cheng Yeh. Default of Credit Card Clients. UCI Machine Learning Repository, 2009. DOI: <https://doi.org/10.24432/C55S3H>.
- [59] Tianyu Zhang, Shen Dong, O. Deniz Kose, Yanning Shen, and Yupeng Zhang. FairZK: A Scalable System to Prove Machine Learning Fairness in Zero-Knowledge . In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 3460–3478, Los Alamitos, CA, USA, May 2025. IEEE Computer Society. doi: 10.1109/SP61157.2025.00205. URL <https://doi.ieeecomputersociety.org/10.1109/SP61157.2025.00205>.

A Open Science

Our code, data and model parameters can be found at: <https://github.com/sankha555/azkaban>. We used Gemini and ChatGPT to help typeset tables in LaTeX. We used Claude Code to help with scripting and code packaging.

B Ethical Considerations

This paper introduces a system to certify deep neural network (DNN) properties while preserving privacy of proprietary model parameters. This enables multiple stakeholders like AI companies, auditors and users to ascertain model reliability. This improves trust in safety-critical sectors like healthcare and finance without risking intellectual property loss. We identified critical soundness vulnerabilities in existing ZK fairness certification protocols stemming from rounding errors in fixed-point arithmetic. Our sound interval analysis for fixed-point arithmetic effectively patches these

gaps and provides guidance on future development. We note that while publishing these findings can expose attack strategies which can be exploited by malicious actors, we judge the benefits of such dissemination to outweigh its risks. Our identified vulnerabilities do not imply immediate exploitation of real-world ML systems as the underlying certification algorithms are not yet deployed in practice. Finally, our work builds entirely on public datasets and honors responsible disclosure.

C Numerical Examples and Attack Proofs

Building on our discussion in Section 5, we present proofs and numerical examples illustrating the identified attack strategies.

C.1 Proof of Over-Estimation by Rounding Up

We give a proof of over-estimation of the true projected distance if the prover consistently rounds up all operations. The prover inputs $\mathbf{a} \in \mathbb{R}^n$ and b and the verifier enters $\mathbf{x} \in \mathbb{R}^n$. Denote $\forall i \in [n] : \bar{a}_i = \lceil a_i \cdot 2^s \rceil$, $\bar{x}_i = \lceil x_i \cdot 2^s \rceil$ and $\bar{b} = \lceil b \rceil$. The projected distance is computed as $D = \left\lceil \left| \sum_i \bar{a}_i \cdot \bar{x}_i + \bar{b} \cdot 2^s \right| / \left\lceil \sqrt{\sum_i \bar{a}_i^2} \right\rceil \right\rceil$. Assuming $\forall i \in [n] : a_i \cdot 2^s \in \mathbb{Z}$ and that $\sum_{i \in [n]} \bar{a}_i$ is a perfect square, we have

$$\begin{aligned}
D &= \left\lceil \left| \sum_i \bar{a}_i \cdot \bar{x}_i + \bar{b} \cdot 2^s \right| / \left\lceil \sqrt{\sum_i \bar{a}_i^2} \right\rceil \right\rceil \\
&\geq \left| \sum_i \bar{a}_i \cdot \bar{x}_i + \bar{b} \cdot 2^s \right| / \left\lceil \sqrt{\sum_i \bar{a}_i^2} \right\rceil \\
&\geq \left| \sum_i \bar{a}_i \cdot \bar{x}_i + \bar{b} \cdot 2^s \right| / \sqrt{\sum_i \bar{a}_i^2} \quad \left(\sum_{i \in [n]} \bar{a}_i \text{ is a perfect square} \right) \\
&\geq \left| \sum_i \bar{a}_i \cdot \bar{x}_i + \bar{b} \cdot 2^s \right| / \sqrt{\sum_i (a_i \cdot 2^s) \cdot (a_i \cdot 2^s)} \quad (a_i \cdot 2^s \in \mathbb{Z}) \\
&\geq \left| \sum_i \bar{a}_i \cdot \bar{x}_i + \bar{b} \cdot 2^s \right| / (2^s \cdot \|\mathbf{a}\|_2) \\
&\geq \left| \sum_i a_i \cdot x_i \cdot 2^{2s} + b \cdot 2^{2s} \right| / (2^s \cdot \|\mathbf{a}\|_2) \quad (\text{definition of ceiling}) \\
&\geq (| \sum_i a_i \cdot x_i + b | / \|\mathbf{a}\|_2) \cdot 2^s
\end{aligned}$$

Thus, the computed distance D in fixed-point always over-estimates the real-valued projected distance. $\square$

C.2 Overestimation by Rounding Down

We present an example of how a malicious prover can over-estimate the projected distance computation by consistently rounding down all fixed-point operations. Consider the prover has a

model such that $\mathbf{a} = (0.8672, 0.0782)$ and $b = 0$. The verifier queries the model on input $\mathbf{x}^* = (1, 1)$. Using a fixed-point scale of $s = 8$ and performing fixed-point arithmetic while always rounding down, the prover publishes the fairness metric as $D = \lfloor \lfloor \underline{\mathbf{a}} \cdot \underline{\mathbf{x}} + \underline{b} \cdot 2^s \rfloor / \lfloor \|\underline{\mathbf{a}}\|_2 \rfloor \rfloor = \lfloor \lfloor 222 \cdot 256 + 20 \cdot 256 + 0 \rfloor / \lfloor \sqrt{222^2 + 20^2} \rfloor \rfloor = 279$. Scaling to reals, the verifier sees and accepts a fairness certificate where $d = 279/2^8 = 1.0898$.

However, the real fairness metric is computed as $d_{\mathbb{R}} = |0.8672+0.0782|/\sqrt{0.8672^2 + 0.0782^2} = 1.0858$. Thus, the prover certifies a false fairness value that is larger by 0.004, i.e., off in the third decimal place. Note that this is larger than the minimum representable number with scale $s = 8$ (≈ 0.0039).

D Deferred Proofs

Proof of Lemma 1 . From soundness of the intervals, $\underline{a} \leq a \cdot 2^s \leq \bar{a}$ and $\underline{b} \leq b \cdot 2^s \leq \bar{b}$. Adding the two inequalities, we obtain $\underline{a} + \underline{b} \leq a \cdot 2^s + b \cdot 2^s \leq \bar{a} + \bar{b} \implies \underline{a} + \underline{b} \leq (a + b) \cdot 2^s \leq \bar{a} + \bar{b} \implies c \cdot 2^s \in [\underline{a} + \underline{b}, \bar{a} + \bar{b}]$.

For proving soundness of subtraction, we multiply the second inequality above by negative one throughout, thus obtaining $-\bar{b} \leq -b \cdot 2^s \leq -\underline{b}$. Adding the latter to the first inequality in a yields $\underline{a} - \bar{b} \leq (a - b) \cdot 2^s \leq \bar{a} - \underline{b} \implies c \cdot 2^s \in [\underline{a} - \bar{b}, \bar{a} - \underline{b}]$. $\square$

Proof of Lemma 2 . Consider two real-valued vectors $\mathbf{a}$ and $\mathbf{b}$ of length n . Denote their inner-product by $c = \sum_{i=1}^n a_i b_i$. By definition, $\forall i \in [n], \underline{a_i} \leq a_i \cdot 2^s \leq \bar{a_i}$ and $\underline{b_i} \leq b_i \cdot 2^s \leq \bar{b_i}$. Then, from interval multiplication we have $\min\{\underline{a_i b_i}, \underline{a_i} \bar{b_i}, \bar{a_i} \underline{b_i}, \bar{a_i} \bar{b_i}\} \leq a_i b_i \cdot 2^{2s} \leq \max\{\underline{a_i b_i}, \underline{a_i} \bar{b_i}, \bar{a_i} \underline{b_i}, \bar{a_i} \bar{b_i}\}$. Thus,

$$\begin{aligned} \underline{C} &= \sum_{i=1}^n \min\{\underline{a_i b_i}, \underline{a_i} \bar{b_i}, \bar{a_i} \underline{b_i}, \bar{a_i} \bar{b_i}\} \leq \sum_{i=1}^n a_i b_i \cdot 2^{2s} \\ \implies \left[2^{-s} \cdot \sum_{i=1}^n \min\{\underline{a_i b_i}, \underline{a_i} \bar{b_i}, \bar{a_i} \underline{b_i}, \bar{a_i} \bar{b_i}\} \right] &\leq 2^s \cdot \sum_{i=1}^n a_i b_i \end{aligned}$$

Similarly, $\bar{C} = 2^s \cdot \sum_{i=1}^n a_i b_i \leq \lceil 2^{-s} \cdot \sum_{i=1}^n \max\{\underline{a_i b_i}, \underline{a_i} \bar{b_i}, \bar{a_i} \underline{b_i}, \bar{a_i} \bar{b_i}\} \rceil$. Combining the two inequalities, we obtain that $\underline{C} \leq c \cdot 2^s \leq \bar{C}$. $\square$

Proof of Lemma 3 By definition, $\underline{a} \leq a \cdot 2^s \leq \bar{a}$ and $\underline{\beta} \leq (\frac{1}{b}) \cdot 2^s \leq \bar{\beta}$. Since both a and b are assumed to be positive, $\underline{a} > 0$ and $\underline{\beta} > 0$. Thus, multiplying the two inequalities yields $\underline{a} \cdot \underline{\beta} \leq \frac{a}{b} \cdot 2^{2s} \leq \bar{a} \cdot \bar{\beta} \implies (\underline{a} \cdot \underline{\beta}) \cdot 2^{-s} \leq \frac{a}{b} \cdot 2^s \leq (\bar{a} \cdot \bar{\beta}) \cdot 2^{-s}$. From the definition of floor, $\lfloor (\underline{a} \cdot \underline{\beta}) \cdot 2^{-s} \rfloor \leq \frac{a}{b} \cdot 2^s$. Similarly, $\frac{a}{b} \cdot 2^s \leq \lceil (\bar{a} \cdot \bar{\beta}) \cdot 2^{-s} \rceil$. Combining the two inequalities yields $c \cdot 2^s = \frac{a}{b} \cdot 2^s \in C$. $\square$

Proof of Lemma 4 We first prove the sound underestimation of the real reciprocal due to Tangent division. Define $x_{\text{norm}} = x \cdot 2^{s-k} \in (1 - 2^{-k}, 2)$ to be the normalized representation of the real number x , where the -2^{-k} term is attributed to rounding errors in the normalization. Since

$\bar{x} \triangleq \lceil x \cdot 2^s \rceil$, we have that $z = \hat{x} \cdot 2^{n-k-1} \geq x \cdot 2^{s+n-k-1}$. Then,

$$\begin{aligned} z &= 2^{n-1} + 2^{n-m-1} \cdot z_2 + z_1 \geq x_{\text{norm}} \cdot 2^{n-1} \\ \implies z_1 &\geq x_{\text{norm}} \cdot 2^{n-1} - 2^{n-1}(1 + 2^{-m} \cdot z_2) \\ \implies z_1 &\geq x_{\text{norm}} \cdot 2^{n-1} - 2^{n-1} \cdot q. \end{aligned}$$

Thus, from line 13 in y' equals

$$\left\lfloor \frac{2^{s+n-1}}{q} \right\rfloor - \left\lfloor \frac{2^s}{q^2} \right\rfloor \cdot z_1 \leq \frac{2^{s+n-1}}{q} - \frac{2^s}{q^2} \cdot z_1 \leq \frac{2^{s+n-1}}{q} - \frac{2^{s+n-1}}{q^2} \cdot (x_{\text{norm}} - q)$$

over reals. Computing $\underline{y}$ using the specified rounding, we have

$$\underline{y} \leq 2^s \cdot 2^{s-k} \cdot \left(\frac{1}{q} - \frac{x_{\text{norm}} - q}{q^2} \right).$$

Since the tangent to a convex function is a global underestimator and the line $y^T = \frac{1}{q} - \frac{x-q}{q^2}$ is tangent to $1/x$ at the point $x = q$, y^T lies below $1/x$ for all reals. Thus, $\frac{1}{q} - \frac{x_{\text{norm}} - q}{q^2} \leq \frac{1}{x_{\text{norm}}}$. Thus, using the definition of x_{norm} , we get $\underline{y} \leq 2^s \cdot \frac{2^{s-k}}{x_{\text{norm}}} = \frac{2^s}{x}$. We now prove sound overestimation of the real reciprocal due to secant division. Define $x_{\text{norm}} = x \cdot 2^{s-k} \in [1, 2 + 2^{-k})$, where the error term 2^{-k} accounts for rounding errors in the normalization. Using similar arguments as the proof for the underestimation above, we start with $\underline{x} \triangleq \lfloor x \cdot 2^s \rfloor \implies z_1 \leq x_{\text{norm}} \cdot 2^{n-1} - 2^{n-1} \cdot q$. Then, from line 13 in Algorithm 2,

$$y' = \left\lceil \frac{2^{s+n-1}}{q} \right\rceil - \left\lceil \frac{2^s}{q^2} \right\rceil \cdot z_1 \geq \frac{2^{s+n-1}}{q} - \frac{2^{s+n-1}}{q(q+\epsilon)} \cdot (x_{\text{norm}} - q)$$

Computing the consequent scaling and denormalization steps,

$$\bar{y} \geq 2^s \cdot 2^{s-k} \cdot \left(\frac{1}{q} - \frac{x_{\text{norm}} - q}{q(q+\epsilon)} \right) \geq 2^s \cdot \frac{2^{s-k}}{x_{\text{norm}}} = \frac{2^s}{x}.$$

□

We remark here that our secant estimator operates over the domain $[2^{-s}, 2^{n-s})$ to ensure the normalization steps are well-defined. This truncated domain still suffices for our abstract interpretation use case. Since division is only used in the ReLU abstract transformer to compute $\frac{u}{u-l}$ where $u > 0$ and $\ell < 0$. In the fixed-point (i.e., integral) domain, the denominator $X = U - L \geq 2 \implies$ over reals, $x = \frac{X}{2^s} \geq \frac{2}{2^s} \geq 2^{-s}$.

E Formal ZKP Specifications

We present our protocol for ZK DNN abstraction and helper protocols for zonotope concretization and minmax computation here.

E.1 ZK Zonotope Concretization

The ZKP for the zonotope concretization is presented in 3. Since our system is based on sound interval analysis over fixed-point arithmetic, we replace the real-valued coefficients in equation 7 with the corresponding fixed-point intervals. Let the variable v_j be abstracted by $V_j^{\mathcal{Z}} = [\underline{A}_{j0}, \overline{A}_{j0}] \oplus \sum_{k=1}^t [\underline{A}_{jk}, \overline{A}_{jk}] \odot \epsilon_k$. Then, the maximum possible perturbation in any direction k is given by the maximum of the absolute endpoints of coefficient $[\underline{A}_{jk}, \overline{A}_{jk}]$. Thus, the center coefficient can be perturbed by at most

$$M = \sum_{k=1}^t \max \left(\left| \underline{A}_{jk} \right|, \left| \overline{A}_{jk} \right| \right).$$

We thus obtain the concrete bounds of v_j as

$$[\ell_j, u_j] = [\underline{A}_{j0} - M, \overline{A}_{j0} + M].$$

In Algorithm 3, step 7 computes the maximum value among the absolute end-points of each noise coefficient. Essentially, step 7 computes the maximum absolute endpoint by multiplexing between the lower-bounding and upper-bounding endpoints based on their signs and absolute value. This is realized by the following logic: if the coefficient is purely positive, we select the upper-endpoint; if the coefficient is purely negative, we select negation of the lower-endpoint; otherwise, we explicitly find the maximum absolute endpoint by multiplexing on the sign of $(\overline{A}_{jk} + \underline{A}_{jk})$. P and N compute in step 6 compute the signs of the endpoints in zero-knowledge using LUT-based comparison [25].

Algorithm 3 Zonotope Concretization

```

1: Input: Zonotope  $V_j^{\mathcal{Z}}$ 
2: Output: Concrete bounds of the variable  $v_j$ .
3:  $\mathcal{N}_j \leftarrow \{k \mid \epsilon_k \in V_j^{\mathcal{Z}}\}$ 
4:  $\ell \leftarrow \underline{A}_{j0}, u \leftarrow \overline{A}_{j0}$ 
5: for  $k \in \mathcal{N}$  do
6:    $\mathcal{P}$  and  $\mathcal{V}$  compute  $P \leftarrow \mathbf{1}\{\underline{A}_{jk} \geq 0\}$  and  $N \leftarrow \mathbf{1}\{\overline{A}_{jk} \leq 0\}$ 
7:    $\mathcal{P}$  and  $\mathcal{V}$  compute  $m \leftarrow P \cdot \overline{A}_{jk} - N \cdot \underline{A}_{jk} + (1 - P) \cdot (1 - N) \cdot \{\mathbf{1}\overline{A}_{jk} + \underline{A}_{jk} \geq 0 \cdot (\overline{A}_{jk} + \underline{A}_{jk}) - \underline{A}_{jk}\}$ 
8:    $\ell \leftarrow \ell - m$ 
9:    $u \leftarrow u + m$ 
10: end for
11: return  $(\ell, u)$ 

```

E.2 ZK Minmax Computation

We present our protocol to simultaneously compute minimum and maximum of four authenticated field elements which we use in our sound interval multiplication and inner-products. Simultaneously computing minimum and maximum allows reuse of comparison results which are expensive to compute in zero-knowledge.

Algorithm 4 Compute min and max of 4 field elements

```
1: Input:  $\tau_0, \tau_1, \tau_2, \tau_3 \in \mathbb{F}_p^4$ 
2: Output:  $\min\{\tau_0, \tau_1, \tau_2, \tau_3\}, \max\{\tau_0, \tau_1, \tau_2, \tau_3\}$ 
3: Compute the Max. of the first two and last two elements
4:  $M_0 \leftarrow \mathbf{1}\{\tau_0 \geq \tau_1\} \cdot (\tau_0 - \tau_1) + \tau_1$ 
5:  $M_1 \leftarrow \mathbf{1}\{\tau_2 \geq \tau_3\} \cdot (\tau_2 - \tau_3) + \tau_3$ 
6: Min. of each pair is the element other than the Max.
7:  $m_0 \leftarrow \tau_0 + \tau_1 - M_0$ 
8:  $m_1 \leftarrow \tau_2 + \tau_3 - M_1$ 
9: Global Max. is greater of the first two maxima
10:  $M \leftarrow \mathbf{1}\{M_0 \geq M_1\} \cdot (M_0 - M_1) + M_1$ 
11: Global Min. is lesser of the first two minima
12:  $m \leftarrow \mathbf{1}\{m_1 \geq m_0\} \cdot (m_0 - m_1) + m_1$ 
13: return  $m, M$ 
```

E.3 ZK DNN Abstract Interpretation

We present the end-to-end ZKP for DNN abstract interpretation that composes the abstract transformers from Section 7. The ZKP proves correct computation of arithmetic circuits defined over $\mathbb{F}_p$ that emulate the field-interval operations of each abstract transformer over IT-MAC commitments of parameters and inputs. The notation used below follows from Section 7. Below we abuse notation by using $\epsilon_k \in V_j^{\mathbb{Z}}$ to denote that the noise symbol ϵ_k appears in the affine expression of variable v_j .

Most steps in Algorithm 5 are straightforward translations of the affine expressions over intervals from Section 7 over finite field intervals authenticated using IT-MACs. Similar to Algorithm 3, the affine expression for the ReLU variables in Algorithm 5 is computed by multiplexing on the signs of the concrete interval of the previous variable v_j . These signs are computed using LUT-based comparisons and represented as intervals P and N (see line 20). Even though the comparison outputs themselves are field elements, we convert them into field intervals for the subsequent multiplexing operations in lines 21, 22 and 25.

The interval division operations in lines 18 and 19 use our sound interval reciprocal operators. The reciprocal endpoints are computed using the tangent and secant approximations in Algorithm 1 and 2, respectively. The coefficients for the respective algorithms are stored in lookup-tables T_{Tangent} and T_{Secant} . Proofs of the table lookup and truncation steps of 1 and 2 are handled by the protocols of [25, 57].

Algorithm 5 Zero-Knowledge DNN Abstract Interpretation

```

1: Public Input: Input vector  $\mathbf{x} \in \mathbb{R}^d$ , certification radius  $\delta \in \mathbb{R}$ , DNN architecture, ground truth  $g \in [o]$ 
2: Assumption: The DNN is expanded into variables  $\{v_i\}_{i=1}^n$  (see Section 7).  $\mathcal{P}$  and  $\mathcal{V}$  have authenticated the parameters for all affine variables, i.e.,  $\mathbf{W}_i = (W_{i1}, \dots, W_{im})$  and  $B_i$ , where  $W_{ij} = [\underline{W}_{ij}, \overline{W}_{ij}]$  and  $B_i = [\underline{b}_i, \overline{b}_i]$  for the affine variable  $v_i$ .  $\mathcal{P}$  and  $\mathcal{V}$  have setup authenticated look-up tables  $T_{\text{Tangent}}$  and  $T_{\text{Secant}}$  to be used for sound interval reciprocal.
3: Output:  $\top$  if input is certified safe, else  $\perp$ 
4: Initialize  $\mathcal{N} \leftarrow \emptyset$  and  $t \leftarrow 0$ 
5:  $\mathcal{P}$  and  $\mathcal{V}$  authenticate  $\{X_j\}_{j \in [d]}$ , where  $X_i = [\underline{x}_i, \overline{x}_i]$ 
6:  $\mathcal{P}$  and  $\mathcal{V}$  authenticate  $\Delta$ , where  $\Delta = [\underline{\delta}, \overline{\delta}]$ 
7: for  $i = 1$  to  $n$  do
8:   if  $v_i.\text{type} = \text{Input}$  then
9:      $\mathcal{P}$  and  $\mathcal{V}$  compute  $V_i^{\mathcal{Z}} \triangleq X_i + \Delta \odot \epsilon_i$ 
10:     $\mathcal{N} \leftarrow \mathcal{N} \cup \{i\}$ 
11:   else if  $v_i.\text{type} = \text{Affine}$  then
12:      $\mathcal{P}$  and  $\mathcal{V}$  validate that  $\mathbf{1}\{\overline{W}_{ij} \geq \underline{W}_{ij}\}$  for all  $j \in [m]$  and  $\mathbf{1}\{\overline{B}_i \geq \underline{B}_i\}$ 
13:      $\mathcal{N}^{\text{prev}} \leftarrow \{k \mid \epsilon_k \in \{V_j^{\mathcal{Z}}\}_{j \in I}\}$  where  $I$  indexes the previous layer's variables (i.e.,  $\mathcal{N}^{\text{prev}}$  is the collection of all previous layer's noise symbol indices)
14:      $\mathcal{P}$  and  $\mathcal{V}$  compute
      
$$V_i^{\mathcal{Z}} \triangleq \underbrace{(B_i \oplus \langle \mathbf{W}_i, \mathbf{A}_0 \rangle)}_{A_{i0}} \oplus \sum_{k \in \mathcal{N}^{\text{prev}}} \underbrace{\langle \mathbf{W}_i, \mathbf{A}_k \rangle}_{A_{ik}} \odot \epsilon_k$$

15:   else if  $v_i.\text{type} = \text{ReLU}$  then
16:     Initialize  $j \in [n]$  such that  $v_i = \max(0, v_j)$ 
17:      $\mathcal{P}$  and  $\mathcal{V}$  compute  $[\ell_j, u_j] \leftarrow \text{concretize}(V_j^{\mathcal{Z}})$ 
18:      $\Lambda \leftarrow [u_j, u_j] \odot ([u_j, u_j] \oplus [\ell_j, \ell_j])$ 
19:      $\Gamma \leftarrow ([u_j, u_j] \otimes ([\ell_j, \ell_j] \otimes [-2^s, -2^s])) \odot ([2^{s+1}, 2^{s+1}] \otimes ([u_j, u_j] \oplus [\ell_j, \ell_j]))$ 
20:      $\mathcal{P}$  and  $\mathcal{V}$  compute  $P \leftarrow [\mathbf{1}\{\ell_j \geq 0\}, \mathbf{1}\{\ell_j \geq 0\}]$  and  $N \leftarrow [\mathbf{1}\{u_j \leq 0\}, \mathbf{1}\{u_j \leq 0\}]$ 
21:      $\mathcal{P}$  and  $\mathcal{V}$  compute  $A_{i0} \leftarrow A_{j0} \odot P \oplus (\Lambda \otimes A_{j0} \oplus \Gamma) \odot ([1, 1] \oplus P) \odot ([1, 1] \oplus N)$ 
22:      $\mathcal{N}_j \leftarrow \{k \mid \epsilon_k \in V_j^{\mathcal{Z}}\}$ 
23:      $\mathcal{P}$  and  $\mathcal{V}$  compute  $\forall k \in \mathcal{N}_j: A_{ik} \leftarrow A_{jk} \odot P \oplus A_{jk} \otimes \Lambda \odot ([1, 1] \oplus P) \odot ([1, 1] \oplus N)$ 
24:      $t \leftarrow |\mathcal{N}| + 1$ 
25:      $\mathcal{P}$  and  $\mathcal{V}$  compute  $A_{it} \leftarrow \Gamma \odot ([1, 1] \oplus P) \odot ([1, 1] \oplus N)$ 
26:      $\mathcal{P}$  and  $\mathcal{V}$  define  $V_i^{\mathcal{Z}} \triangleq A_{i0} \oplus \sum_{k \in \mathcal{N}_j} A_{ik} \odot \epsilon_k \oplus A_{it} \cdot \epsilon_t$ 
27:      $\mathcal{N} \leftarrow \mathcal{N} \cup \{t\}$ 
28:   else if  $v_i.\text{type} = \text{Output}$  then
29:     Initialize  $j \in [n]$  such that the output variable  $v_i$  copies the variable  $v_j$ 
30:      $\mathcal{P}$  and  $\mathcal{V}$  compute  $V_i^{\mathcal{Z}} \triangleq V_j^{\mathcal{Z}}$ 
31:   end if
32: end for
33:  $O \leftarrow \{i \mid v_i.\text{type} = \text{Output}\}$  (set of all output variables)
34: Initialize  $G \in O$  such that  $G$  is the  $g$ -th smallest index in  $O$  (i.e., the index corresponding to the ground truth output logit variable)
35:  $\text{safe} \leftarrow 1$ 
36: for  $i \in O$  do
37:    $\mathcal{P}$  and  $\mathcal{V}$  compute  $V_{\text{diff}} \triangleq V_G^{\mathcal{Z}} \ominus V_i^{\mathcal{Z}}$ 
38:    $\mathcal{P}$  and  $\mathcal{V}$  compute  $[\ell, u] \leftarrow \text{concretize}(V_{\text{diff}}^{\mathcal{Z}})$ 
39:    $\mathcal{P}$  and  $\mathcal{V}$  compute  $\text{safe} \leftarrow \text{safe} \cdot \mathbf{1}\{\ell \geq 0\}$ 
40: end for
41: if  $\text{safe}.\text{reveal} = 1$  return  $\top$  else return  $\perp$ 

```
