FairProof: Confidential and Certifiable Fairness for Neural Networks

Chhavi Yadav[1], **Amrita Roy Chowdhury**[1], **Dan Boneh**[2], **Kamalika Chaudhuri**[1]*
[1]University of California, San Diego [2] Stanford University

Abstract

Machine learning models are increasingly used in societal applications, yet legal and privacy concerns demand that they very often be kept confidential. Consequently, there is a growing distrust about the fairness properties of these models in the minds of consumers, who are often at the receiving end of model predictions. To this end, we propose *FairProof*– a system that uses Zero-Knowledge Proofs (a cryptographic primitive) to publicly verify the fairness of a model, while maintaining confidentiality. We also propose a fairness certification algorithm for fully-connected neural networks which is befitting to ZKPs and is used in this system. We implement *FairProof* in Gnark and demonstrate empirically that our system is practically feasible.

1 Introduction

Recent usage of ML models in high-stakes societal applications [37, 13, 17] has raised serious concerns about their fairness [4, 59, 16, 60]. As a result, there is growing distrust in the minds of a consumer at the receiving end of ML-based decisions [20]. To increase consumer trust there is a need for developing technology that enables public verification of the fairness properties of models.

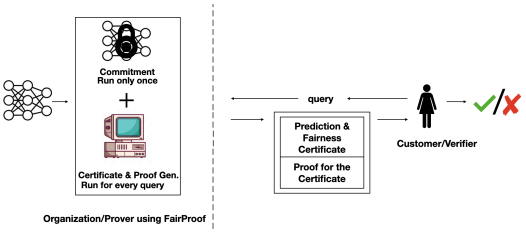


Figure 1: Pictorial Representation of *FairProof*

A major barrier to such verification is that legal and privacy concerns demand that models be kept confidential by organizations. The resulting lack of verifiability can lead to potential misbehavior, such as model swapping, wherein a malicious entity uses different models for different customers leading to unfair behavior. Therefore what is needed is a solution which allows for public verification of the fairness of a model and ensures that the same model is used for every prediction (model uniformity) while maintaining model confidentiality. The canonical approach to evaluating fairness is a statistics-based third-party audit [63, 64, 48, 54]. This approach however is replete with problems arising from the usage of a reference dataset, the need for a trusted third-party, leaking details about the confidential model [14, 28] and lack of guarantees of model uniformity [22, 53].

We address the aforementioned challenges by proposing a system called *FairProof* involving two parts: 1) a fairness certification algorithm which outputs a certificate of fairness , and 2) a cryptographic protocol using commitments and Zero-Knowledge Proofs (ZKPs) that guarantees model uniformity and gives a proof that the certificate is correct.

Given an input query, the fairness certification algorithm outputs how fair the model is at that point *according to a fairness metric*. The metric we use is local Individual Fairness (IF) [19, 32, 8, 9],

*Corresponding author : cyadav@ucsd.edu

2nd Workshop on Regulatable ML at NeurIPS 2024.

which is desirable for two reasons. First, it evaluates fairness of the model at a specific data point (rather than for the entire input space) – this allows us to give a *personalized* certificate to every customer, as would be required by customer-facing organizations. Second, it works on the model post-training, making it completely *agnostic* to the training pipeline.

How do we design a certification algorithm for the chosen metric? We observe that certifying local IF can be reduced to an instantiation of certifying robustness.[2] We then leverage techniques from the robustness literature to design our algorithm. One of our key contributions is to design the algorithm so that it is ZKP-friendly. In particular, the computational overhead for ZKPs depends on the complexity of the statement being proved. To this end, we design a fairness certificate which results in relatively low complexity statements.

Once fairness certificate has been computed, we want to enable the consumer to verify that the certificate was indeed computed correctly, but without revealing the model weights. To do this, we rely on Succinct Zero Knowledge Proofs [26, 25]. This cryptographic primitive enables a prover (eg. bank) to prove statements (eg. fairness certificate) about its private data (eg. model weights) without revealing the private data itself. It provides a proof of correctness as an output. Then a verifier (eg. customer) verifies this proof without access to the private data. In our case, if the proof passes verification, it implies that the fairness certificate was computed correctly w.r.t the hidden model.

We design and implement a specialized ZKP protocol to efficiently prove and verify the aforementioned fairness certification algorithm. Doing this naively would be very computationally expensive. We tackle this challenge with three insights. First, we show that verification of the entire certification algorithm can be reduced to a few strategically chosen sub-functionalities, each of which can be proved and verified efficiently. Second, we provide a *lower* bound on the certificate, i.e., a conservative estimate of the model's fairness, for performance optimization. Third, we observe that certain computations can be done in an offline phase thereby reducing the online computational overhead.

Our solution ensures model uniformity through standard cryptographic commitments. A cryptographic commitment to the model weights binds the organization to those weights publicly while maintaining confidentiality of the weights. This has been widely studied in the ML security literature [27, 11, 34, 41, 55, 46, 45].

Experiments. In this work we focus on fully-connected neural networks with ReLU activations as the models. We implement and evaluate *FairProof* on three standard fairness benchmark datasets to demonstrate its practical feasibility. For instance, for the *German* [30] dataset, we observe that *FairProof* takes around 1.17 minutes on an average to generate a verifiable fairness certificate per data point without parallelism or multi-threading on an Intel-i9 CPU chip. The communication cost is also low – the size of the verifiable certificate is only 43.5KB.

2 Preliminaries & Setting

Fairness. Existing literature has put forth a wide variety of fairness definitions [44, 5]. In this paper, we focus on the notion of *local individual fairness* [32, 19, 8] defined below, as it best aligns with our application (see Sec. 2 for more details).

Definition 1 (Local Individual Fairness)**.** *A machine learning model $f : \mathbb{R}^n \mapsto \mathcal{Y}$ is defined to be ϵ-individually fair w.r.t to a data point $x^* \sim \mathcal{D}$ under some distance metric $d : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ if*

$$\forall x : \; d(x, x^*) \leq \epsilon \implies f(x^*) = f(x) \tag{1}$$

We say a model f is exactly ϵ^*-individually fair w.r.t x^* if ϵ^* is the largest value that satisfies Eq. 1. In particular, ϵ^* is known as the local individual fairness parameter. **For brevity we will be using ϵ to mean ϵ^* and fairness/individual fairness to refer to the notion of local individual fairness, unless stated otherwise, throughout the rest of the paper.**

Individual fairness formalizes the notion that similar individuals should be treated similarly; more precisely, get the same classification. The similarity is defined according to a task dependent distance metric $d(\cdot)$ that can be provided by a domain expert. Examples of such a metric could be weighted ℓ_p norm where the weights of the sensitive features (race, gender) are set to zero [8].

[2]Certifiable Robustness quantifies a model's resistance to adversarial attacks by measuring the extent to which a data point can be perturbed without altering the model prediction.

Neural Networks. We focus on the classification task and consider neural network (NN) classifiers $f : \mathcal{X} \mapsto \mathcal{Y}$, where f is a fully-connected neural network with ReLU activations, $\mathcal{X} = \mathbb{R}^n$ is the input space and $\mathcal{Y}$ is a discrete label set. This NN classifier (pre-softmax) can also be viewed a collection of piecewise linear functions over a union of convex polytopes [62, 29, 49, 15, 52]. Here each linear function corresponds to one polytope and each polytope corresponds to one activation pattern of the nodes in the NN. A polytope $\mathcal{P}$ is represented by a set of linear inequalities, $\mathcal{P} = \{x | \mathbf{A}x \leq \mathbf{b}\}$; then the collection of all such polytopes forms a partition of the input domain, $\mathcal{X} = \bigcup \mathcal{P}$ (App. Fig. 6).

A facet is an $(n-1)$-*face* of the polytope corresponding to the set $\{x | x \in \mathcal{P} \cap \mathbf{A}_i x = \mathbf{b}_i\}$ where $\mathbf{A}_i$ and $\mathbf{b}_i$ are the values of $\mathbf{A}$ and $\mathbf{b}$ at the i^{th} dimension. Two polytopes that share a facet are known as neighboring polytopes. The decision region of f at a data point x^* is defined as the set of points for which the classifier returns the same label as it does for x^*, essentially the set $\{x | f(x) = f(x^*)\}$. This decision region can also be expressed as a union of convex polytopes [33]. A facet that coincides with the decision boundary of f is known as a boundary facet. See App. A for details.

Cryptographic Primitives. We use two cryptographic primitives, namely commitment schemes and zero knowledge proof, for verifying the individual fairness certification.

A *Commitment Scheme* commits to a private input w without revealing anything about w; its output is a commitment string com_w. A commitment scheme has two properties:

1. *Hiding*: the commitment string com_w reveals nothing about the committed value w.
2. *Binding*: it is not possible to come up with another input w' with the same commitment string as w, thus binding w to com_w (simplified).

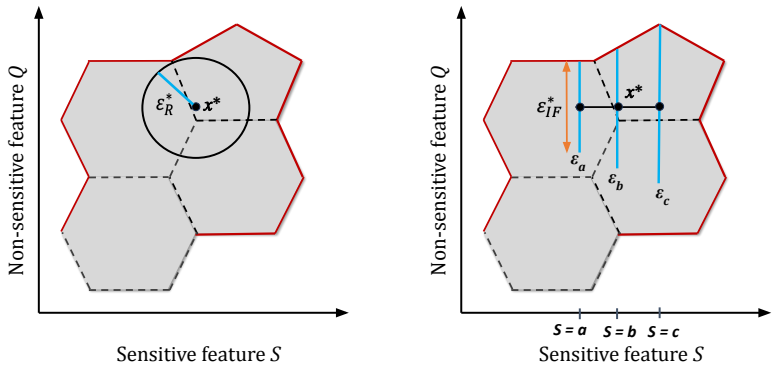
Zero Knowledge Proofs [26] describe a protocol between two parties – a prover and a verifier, who both have access to a circuit P. A ZKP protocol enables the prover to convince the verifier that it *possesses* an input w such that $P(w) = 1$, without revealing any additional information about w to the verifier. A simple example is when $P_\varphi(w) = 1$ iff φ is a SAT formula and $\varphi(w) = 1$; a ZKP protocol enables the prover to convince a verifier that there is a w for which $\varphi(w) = 1$, while revealing nothing else about w. A ZKP protocol has the following properties:

1. *Completeness*. For any input w such that $P(w) = 1$, an honest prover who follows the protocol correctly can convince an honest verifier that $P(w) = 1$.
2. *Soundness.* Given an input w that $P(w) \neq 1$, a malicious prover who deviates arbitrarily from the protocol cannot falsely convince an honest verifier that $P(w) = 1$, with more than negligible probability.
3. *Zero knowledge.* If the prover and verifier execute the protocol to prove that $P(w) = 1$, even a malicious verifier, who deviates arbitrarily from the protocol, can learn no additional information about w other than $P(w) = 1$.

Theory suggests that it is possible to employ ZKPs to verify any predicate P in the class NP [25]. Moreover, the resulting proofs are non-interactive and succinct. However, in practice, generating a proof for even moderately complex predicates often incurs significant computational costs. To this end, our main contribution lies in introducing a ZKP-friendly certification algorithm, to facilitate efficient fairness certificate generation.

Problem Setting. A model owner holds a confidential classification model f that cannot be publicly released. User supplies an input query x^* to the model owner, who provides the user with a prediction label $y = f(x^*)$ along with a fairness certificate $\mathscr{C}$ w.r.t to x^*. This certificate can be verified by the user & the user is also guaranteed that the model owner uses the same model for everyone.

The above setting needs three tools. First, the model owner requires an algorithm for generating the fairness certificate with white-box access to the model weights. This algorithm is discussed in Sec. 3. Second, a mechanism is needed that enables the user to *verify* the received certificate (public verification) *without violating model confidentiality*. This mechanism is discussed in Sec. 4. Third, a mechanism is needed to guarantee that the same model is used for everyone (model uniformity), also without violating model confidentiality. For ensuring uniformity, the model owner should commit the model in the initially itself, before it is deployed for users. This has been widely studied and implemented by prior work as discussed in the introduction and an actual implementation of commitments is out of scope of this work.



(a) Certifiable robustness (b) Certifiable fairness

Figure 2: Connection between robustness & fairness for $n = 2$ and one sensitive feature S with values $\{a, b, c\}$. Final fairness certificate is the minimum of $\{\epsilon_a, \epsilon_b, \epsilon_c\}$. Red color denotes decision boundary.

3 How to Certify Individual Fairness?

In this section we present an algorithm to compute a local individual fairness certificate. This certificate is computed by the model owner with white-box access to the model weights and is specific to each user query, thereby leading to a *personalized* certificate. The certificate guarantees to the user that the model has certain fairness properties at their specific query.

Preliminaries. Starting with some notation, let $\mathcal{S}$ be the set of k sensitive features, $\mathcal{S} := \{S_1, \cdots, S_k\}$ where S_i denotes the i^{th} sensitive feature. We assume that each sensitive feature S_i has a discrete and finite domain, denoted by $domain(S_i)$, which is in line with typical sensitive features in practice, such as race (eg. black/white/asian), presence of a medical condition (yes/no). Let $domain(\mathcal{S})$ represent the set of all possible combinations of the values of sensitive features, $domain(\mathcal{S}) := domain(S_1) \times \cdots \times domain(S_k)$. Without loss of generality, any data point $x \in \mathbb{R}^n$ is also represented as $x = x_{\setminus\mathcal{S}} \cup x_{\mathcal{S}}$, where $x_{\setminus\mathcal{S}}$ and $x_{\mathcal{S}}$ are the non-sensitive and sensitive features of x.

For the distance metric in individual fairness (Eq. 1), we consider a weighted ℓ_2-norm where the non-sensitive features have weight 1 while the sensitive features have weight 0. This distance metric is equivalent to the ℓ_2-norm sans the sensitive features. Thus, based on Def. 1, f is ϵ-individually fair w.r.t x^* iff,

$$\forall x : ||x_{\setminus\mathcal{S}} - x^*_{\setminus\mathcal{S}}||_2 \leq \epsilon \implies f(x^*) = f(x) \tag{2}$$

With this notation in place, observe that our fairness certificate $\mathscr{C}$ is essentially the value of the parameter ϵ. Intuitively it means that the model's classification is independent of the sensitive features as long as the non-sensitive features lie within an ℓ_2 ball of radius ϵ centered at $x^*_{\setminus\mathcal{S}}$. Eq.2 can also be equivalently viewed as follows: set the sensitive features of x^* and x to a particular value $s \in domain(\mathcal{S})$ (so that they cancel out in the norm), then find the corresponding certificate ϵ_s and repeat this procedure for all values in $domain(\mathcal{S})$; the final certificate ϵ is the minimum of all ϵ_s.

Next we propose an algorithm to compute this fairness certificate. Our algorithm is based on three key ideas, as we describe below.

Idea 1: Reduction from fairness to robustness. Our first key observation is that in our setting, certifiable fairness can be reduced to an instantiation of certifiable robustness, enabling us to re-use ideas from existing robustness literature for our purpose. In particular, the reduction is as follows. A model f is defined to be ϵ-pointwise ℓ_2 robust (henceforth robustness) for a data point x^*, if

$$\forall x : ||x - x^*||_2 \leq \epsilon \implies f(x^*) = f(x) \tag{3}$$

Comparing this definition to Eq.2 and its alternate view, we observe that once the sensitive features have been fixed to a value $s \in domain(\mathcal{S})$, computing the corresponding fairness certificate ϵ_s is *equivalent* to solving the robustness problem in $(n - k)$ dimensions where the k dimensions

corresponding to the sensitive features $\mathcal{S}$ are excluded. Let us assume there exists an algorithm which returns the pointwise ℓ_2 robustness value for an input. Then the final fairness certificate ϵ computation requires $|domain(\mathcal{S})|$ calls to this algorithm, one for each possible value of the sensitive features in $\mathcal{S}$. Fig. 2 illustrates this idea pictorially for NNs.

For ReLU-activated neural networks represented using n-dimensional polytopes, setting the values of sensitive features implies bringing down the polytopes to $(n-k)$ dimensions. Geometrically, this can be thought of as slicing the n-dimensional polytopes with hyperplanes of the form $x_i = s_i$ where x_i is the i^{th} coordinate, set to the value s_i.

Algorithm 1 Individual Fairness Certification

Inputs $x^* \in \mathbb{R}^n$, f : ReLU-activated Neural Network
Output ϵ_{LB} : Our Fairness Certificate for x^*
1: Construct the set of all polytopes $\mathbb{P} = \bigcup \mathcal{P}$ for f where each polytope is expressed as $\mathcal{P} = \{x | \mathbf{A}x \leq \mathbf{b}\}$
2: $E := [\,]$
3: **for** $s \in domain(S_1) \times \cdots \times domain(S_k)$
4: $\quad \mathbb{P}' :=$ ReducePolyDim($\mathbb{P}$, s) (Alg. 3 in Appendix)
5: $\quad \epsilon_s :=$ GeoCert($x^*, \mathbb{P}', d_{proj}$)
6: $\quad E.append(\epsilon_s)$
7: **end for**
8: $\epsilon_{LB} := \min E$
9: **Return** ϵ_{LB}

Idea 2. Using an efficient certified robustness algorithm. For ReLU-activated neural networks (see Sec.2), the naive algorithm for certifying robustness is infeasible; it entails computing the distance between x^* and *all* boundary facets (facets coinciding with the decision boundary of the model) induced by the model, which is exponential in the number of hidden neurons. Instead, we rely on an efficient iterative algorithm *GeoCert* (Alg. 2 in App. B), proposed by [33]. This algorithm starts from the polytope containing the data point x^* and *iteratively* searches for the boundary facet with the minimum distance from x^*. A priority queue of facets is maintained, sorted according to their distance from x^*. At each iteration, the facet with the minimum distance is popped and its neighbors (polytopes adjacent to this facet) are examined. If the neighboring polytope is previously unexplored, the distance to all of its facets is computed and inserted them into the priority queue; otherwise the next facet is popped. The algorithm terminates as soon as a boundary facet is popped. Fig. 3 presents a pictorial overview of *GeoCert*. Additional details are in App. B.

Idea 3: Generate a lower bound ϵ_{LB} for efficient ZKP. *GeoCert* provides exact fairness certificates ϵ^*, by using a constrained quadratic program solver to get the actual distance between the input point and a facet. However, verifying this solver using ZKPs would be a highly computationally intensive task. Instead we propose to report a lower bound on the certificate, $\epsilon_{LB} < \epsilon^*$, which considerably improves performance. A lower bound means that the reported certificate ϵ_{LB} is a conservative estimate – the true measure of the model's fairness could only be higher. Instead of the exact distance, we compute the *projection* distance between the input point and the *hyperplane* containing the facet (facet is a subset of the hyperplane), which gives a lower bound on the exact distance between x^* and the facet. The projection distance computation involves simple arithmetic operations which are relatively computationally feasible for ZKPs (see Sec. 4 for more details). App. Fig. 7 shows this.

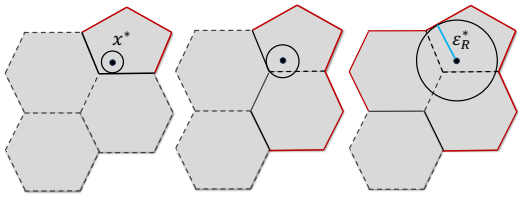


Figure 3: *GeoCert*'s behavior on point x^*. Colored facets are in the priority queue; red and solid black lines denote boundary and non-boundary facets respectively. Algorithm stops when the minimum distance facet is a boundary facet (rightmost).

Theorem 3.1. *Given a data point x^* and a neural network f, Alg. 1 provides a lower bound ϵ_{LB} of the correct individual fairness parameter of x^*.*

Proof for this theorem is given in App. C, Thm. C.3. Our resulting fairness certification algorithm is described in Alg.1 and detailed in App. B.

4 *FairProof*: Verification of the Individual Fairness Certificate

Without careful design choices ZKPs can impose significant computational overhead. To this end, we design an efficient verification protocol named *FairProof* by combining insights from cryptography and ML. Specifically, *FairProof* is based on three key ideas described below.

Idea 1: Strategic verification of sub-functionalities. A naive verification mechanism replicates all the computations outlined in Alg.1. However, this would involve computing *all* the polytopes during *every* proof generation – this is computationally expensive since the number of polytopes is exponential in the number of hidden neurons in the model. In contrast, we show that the verification can be streamlined by focusing on five strategically chosen sub-functionalities, each of which can be checked using certain properties of polytopes and neural networks. Consequently, we only verify the polytopes *traversed* by the certification mechanism.

Idea 2: Representative points. Certain numeric properties of a polytope can be efficiently proven if one has access to a representative point in the interior of the polytope. We leverage this insight in *FairProof* to efficiently verify our chosen sub-functionalities, discussed in the following sections.

Idea 3: Offline computation. We show that certain computations can be performed offline which further reduces the time needed in the online phase.

Next, we detail our verification mechanism *FairProof*. Recall that in our setting model owner is the prover and user is the verifier. The verification consists of two phases:

Phase 1: Pre-processing. All the operations in this phase are executed only once and before the model is deployed to the users. The following two actions need to be taken by the model owner in this phase.

1. Commit to the weights $\mathbf{W}$ of the model f, resulting in the commitment $\mathsf{com}_{\mathbf{W}}$ (we assume that the architecture of f is known, i.e., f is a fully connected neural lnetwork with ReLU activations).
2. Compute a representative point $z_{\mathcal{P}}$ for each polytope $\mathcal{P}$. Additionally, it computes a representative point $z_{\mathcal{F}}$ for every facet $\mathcal{F}$[3][4].

Phase 2: Online verification. The online verification phase is executed *every* time a user submits a query x^* to the model owner for prediction. Verifying the computation of Algorithm 1 essentially amounts to verifying *GeoCert* with some modifications and consists of five steps. The model owner generates proofs for these five functionalities and the user validates them.

1. Verifying initial polytope (Alg. 5). Recall that *GeoCert* starts from the polytope containing data point x^*. Hence, the verifier needs to check that the initial polytope (1) indeed contains the data point x^*, and (2) is one of the polytopes obtained from the model f. The key idea used in this function is that each polytope is associated with a unique ReLU activation code. Verification for step (1) involves computing the ReLU activation code for x^* using the committed weights $\mathsf{com}_{\mathbf{W}}$ and step (2) involves deriving the corresponding polytope for this activation code from $\mathsf{com}_{\mathbf{W}}$.

2. Verifying distance to facets (Alg. 6). During its course *GeoCert* computes distance between x^* and various facets. Hence, the verifier needs to check the correctness of these distance computations. As discussed in the preceding section, we compute a lower bound of the exact distance using projections, which can be efficiently proved under ZKPs.

3. Verifying neighboring polytopes (Alg. 7). In each iteration *GeoCert* visits a neighboring polytope adjacent to the current one; the two polytopes share the facet that was popped in the current iteration. Verifying neighborhood entails checking that the visited polytope indeed (1) comes from the model f, and (2) shares the popped facet. The key idea used here is that two neighboring polytopes differ in a single ReLU activation corresponding to the shared facet (Fact A.2). Specifically, the prover retrieves the representative point corresponding to the visited polytope and computes its ReLU activation code, R', using the committed weights $\mathsf{com}_{\mathbf{W}}$. Next, it computes the polytope corresponding to R' from $\mathsf{com}_{\mathbf{W}}$ to prove that it is obtained from the model f. This is followed by showing that the hamming

[3] A facet is also essentially a polytope, albeit in the $(n-1)$-dimensional space.

[4] Although the number of polytopes and facets are exponential in the number of the neurons in the model, this is a one-time computation performed completely offline and can be parallelized. See Sec. 5 for empirical overhead of this pre-processing step on models for standard datasets.

distance between R' and R is one, where R is the activation code for the current polytope. Finally, the prover shows that the current facet is common to both the polytopes.

4. Verifying boundary facet (Alg. 8). Termination condition of *GeoCert* checks whether the current facet is a boundary facet or not; we verify this in *FairProof* as follows. Let R denote the activation code for the current polytope $\mathcal{P}$ and let $f_R(x) = W_R x + b_R$ represent the linear function associated with R. For the ease of exposition, let f be a binary classifier. In other words, $f_R(x)$ is the input to the softmax function in the final layer of f (i.e., logits) for all data points $x \in \mathcal{P}$. The key idea for verification is that *iff* x lies on a boundary facet, $f_R(x)$ has the *same* value for both the logits. For verifying this computation, we rely on the pre-computed representative point of a facet. Specifically, the prover retrieves the representative point z for the current facet $\mathcal{F} = \{x | Ax \leq b\}$. First, it proves that z lies on $\mathcal{F}$ by showing $Az \leq b$ holds. Next, the prover computes f_R (i.e., the weights W_R and b_R) from the committed weights using R and tests the equality of both the logits in $f_R(z)$.

5. Verify order of facet traversal (Alg. 9). The order in which the facets are traversed needs to be verified – this is equivalent to checking the functionality of the priority queue in *GeoCert*. Standard ZKP tools are built for verifying mathematical computations (expressed as an arithmetic or Boolean circuit) and do not have built-in support for data structures, such as priority queues. We overcome this challenge by leveraging the following key idea – correctness of the priority queue can be verified by checking that the next traversed facet is indeed the one with the shortest distance.

Additional optimizations. We identify certain computations in the above algorithms that can performed offline. Specifically, in *VerifyNeighbor* the proof of correctness for polytope construction using representative points can be generated offline. Further, in *VerifyBoundary* proof for computation of the linear function f_R can also be generated offline. This leads to a significant reduction in the cost of the online proof generation (see Sec. 5).

End-to-end verification mechanism is presented in Alg. 4. In the final step, the prover has to generate an additional proof that the reported certificate of fairness corresponds to the smallest value among all the lower bounds obtained for each element of $domain(\mathcal{S})$ (*VerifyMin* , Alg. 10). Additionally, the prover also needs to prove integrity of the inference, i.e., $y = f(x^*)$. For this, after computing the linear function $f_{R_{x^*}}(x^*)$ using the committed weights $\mathsf{com}_{\mathbf{W}}$ (where R_{x^*} is the activation code for x^*) we need to additionally prove that the label corresponds to the logit with the highest score (Alg. 11, *VerifyInference*). Next, we present our security guarantee.

Theorem 4.1. *(Informal) Given a model f and a data point x^*, FairProof provides the prediction $f(x^*)$ and a lower bound ϵ_{LB} on the individual fairness parameter for x^* without leaking anything, except the number of total facets traversed, about the model f.*

Proof of the above theorem follows directly from the properties of zero-knowledge proofs and theorems in App. D. The formal guarantee and detailed proof is presented in App. D.

5 Evaluation

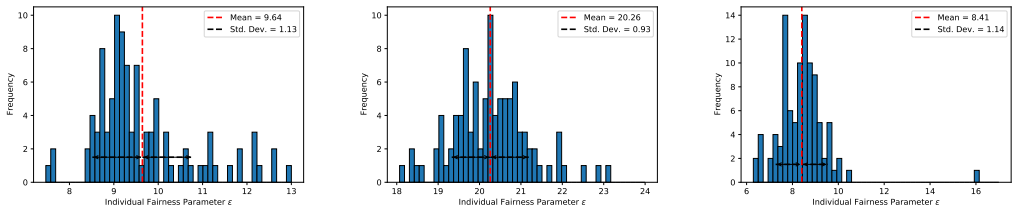
In this section we evaluate the performance of *FairProof* empirically. Specifically, we ask the following questions:

1. Can our fairness certification mechanism distinguish between fair and unfair models?
2. Is *FairProof* practically feasible, in terms of time and communication costs?

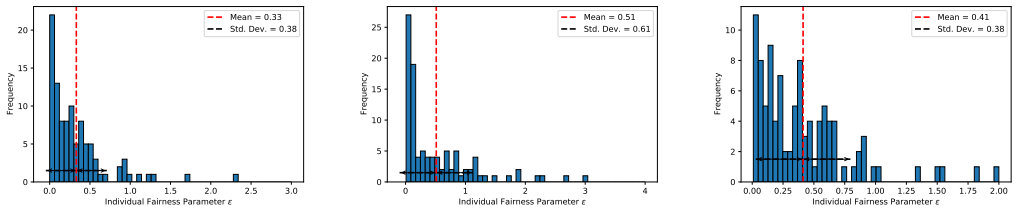
Datasets. We use three standard fairness benchmarks. *Adult [7]* is an income classification dataset, where we select *gender* (male/female) as the sensitive feature. *Default Credit [65]* is a loan defaults prediction dataset, with *gender* (male/female) as the chosen sensitive feature. *German Credit [30]* is a loan application dataset, where *Foreign Worker* (yes/no) is the chosen sensitive feature.

Configuration. We train fully-connected ReLU networks with stochastic gradient descent in PyTorch. Our networks have 2 hidden layers with different sizes including $(4, 2)$, $(2, 4)$ and $(8, 2)$. All the dataset features are standardized [1]. *FairProof* is implemented using the Gnark [12] zk-SNARK library in GoLang. We run all our code for *FairProof* without any multithreading or parallelism, on an Intel-i9 CPU chip with 28 cores.

Model Fairness We first evaluate if our certification mechanism can distinguish between fair and unfair models. Prior work [31] has shown that overfitting leads to more unfair models while regular-



Credit, Fair *Adult, Fair* *German, Fair*

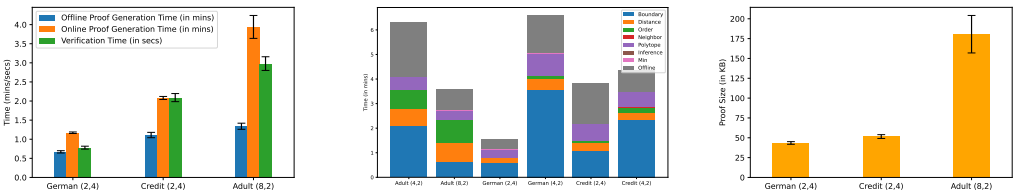


Credit, Unfair *Adult, Unfair* *German, Unfair*

Figure 4: Histogram of fairness parameter ϵ for fair & unfair models, model size = (4,2). ϵ values are higher than those for unfair models.

ization encourages fairness. Thus, to obtain models with different fairness, we vary regularization by changing the weight decay parameter in PyTorch. Then we randomly sample 100 test data points as input queries and find the fairness parameter ϵ for both types of models on these queries.

As demonstrated in Fig. 4, unfair models have a lower ϵ than corresponding fair models. This consistent difference in ϵ values across different model sizes & datasets shows that our certification mechanism can distinguish between fair and unfair models. Results for other models are in App. E.



(a) (b) (c)

Figure 5: (a) Proof Generation (in mins) and Verification times (in secs) for different models. Offline computations are done in the initial setup phase while Online computations are done for every new query. Verification is only done online, for every query. (b) Breakdown of the proof generation time (in mins) for the data point with the median time. (c) Total Proof Size (in KB) for various models. This includes the proof generated during both online and offline phases.

Performance of *FairProof* Since computation is a known bottleneck in ZKPs, we next investigate the overhead of *FairProof* in terms of time and communication costs. All reported numbers are averages over a 100 random test points.

Fig. 5 (a) shows the proof generation costs for various models. Note that the proof generation time varies with the models, due to its dependence on the number of traversed facets[5] which in turn depends on the model and query. On average, the adult model has a larger number of traversed facets than others as shown in Table 1 in App. E, leading to a higher proof generation time. We also observe that performing some computations in an offline phase results in significant reductions in the online time cost, the largest being 1.74×. See Table 1 and Fig.13 in App.E for details.

[5]As mentioned in Thm. 4.1, this information is leaked by *FairProof*.

We also breakdown the overall proof generation time in terms of different sub-functionalities. We report this breakdown for the query with the median proof generation cost, in Fig. 5 (b). As shown in the figure, *VerifyBoundary* is the costliest sub-function for all the models; this is so since it is executed in every iteration (every time a facet is popped) and involves costly non-linear comparison operations (see Alg. 8). Other functionalities that are also executed multiple times based on number of traversed facets but are not as expensive include *VerifyNeighbor*, *VerifyDistance* and *VerifyOrder* (see Alg. 7, 6, 9). The least time is taken by *VerifyMin* which basically finds the minimum in a list; this is so since the function is straight-forward and is ran only once per query (see Alg. 10).

We also report the average verification times - time for checking the validity of the proof by the verifier - in Fig. 5 (a). Note that the verification costs are orders of magnitude lower (in seconds) than the proof generation costs (in minutes) for all models; as is standard in ZKPs. Fig.5 (c) reports the communication overheads, i.e. size of the generated proofs. The proof size is very small, only certain kilobytes. Low verification time and communication cost is advantageous since it implies quick real-time verification which does not require complex machinery at the customer end. For detailed results on all models, refer to Fig. 14 and Fig. 15 in App. E.

Discussion on Scalability For very large models, the number of traversed facets can be huge and running *FairProof* on them may not be practically feasible anymore. In such cases, one solution can be just verifying the fairness of the final layers. We leave this exploration to future work.

6 Related Work

Verifiable fairness with cryptography. Most of the prior work on verifying fairness *while maintaining model confidentiality* [48, 39, 57, 51, 47] has approached the problem in the third-party auditor setting. The closest to ours is a recent work by [53], which proposed a fairness-aware training pipeline for *decision trees* that allows the model owner to cryptographically prove that *the learning algorithm* used to train the model was fair by design. In contrast, we focus on neural networks and issue a fairness certificate by simply inspecting the model weights *post-training*. Our system *FairProof* and certification mechanism is completely agnostic of the training pipeline.

Another line of work has been using cryptographic primitives to verify other properties (rather than fairness) of an ML model while maintaining model confidentiality – [68, 42] focus on accuracy and inference, while [69, 23, 56] focus on the training process.

A separate line of work uses formal verification approaches for verifying the fairness of a model [3, 6, 58, 24, 10]. However, these works focus on certification in the plain text, i.e., they do not preserve model confidentiality and do not involve any cryptography.

Fairness Certification Mechanisms. Prior work on certification mechanisms for fairness can be broadly classified into three categories. The first line of work frames the certification problem as an optimization program [32, 8, 36]. The second line of research has leveraged the connection between robustness and fairness, and proposed fairness-aware training mechanisms akin to adversarial training [50, 67, 38, 66, 18]. In contrast to both, we focus on *local* IF specifically for neural networks and use an iterative algorithm rather than solving a complex optimization problem and are completely agnostic of the training pipeline.

The final line of work is based on black-box query access learning theoretic approaches [63, 64, 43]. Contrary to our work, these approaches however are replete with problems arising from the usage of a reference dataset [22, 53], the need for a trust third-party, and lack of guarantees of model uniformity. See App. Sec. F for a further discussion on related works.

7 Conclusion

In this paper we proposed *FairProof*– a protocol enabling model owners to issue publicly verifiable certificates while ensuring model uniformity and confidentiality. Our experiments demonstrate the practical feasibility of *FairProof* for small neural networks and tabular data. While our work is grounded in fairness and societal applications, we believe that ZKPs are a general-purpose tool and can be a promising solution for overcoming problems arising out of the need for model confidentiality in other areas/applications as well. We call for further research in this direction.

References

[1] Standarization. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

[2] Zator: Verified inference of a 512-layer neural network using recursive snarks. https://github.com/lyronctk/zator/tree/main, 2023.

[3] Aws Albarghouthi, Loris D'Antoni, Samuel Drews, and Aditya V. Nori. Fairsquare: Probabilistic verification of program fairness. *Proc. ACM Program. Lang.*, 1(OOPSLA), oct 2017.

[4] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing, 2016.

[5] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning: Limitations and Opportunities*. fairmlbook.org, 2019. http://www.fairmlbook.org.

[6] Osbert Bastani, Xin Zhang, and Armando Solar-Lezama. Probabilistic verification of fairness properties via concentration. *Proc. ACM Program. Lang.*, 3(OOPSLA), oct 2019.

[7] Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

[8] Elias Benussi, Andrea Patané, Matthew Wicker, Luca Laurenti, Marta Kwiatkowska University of Oxford, and Tu Delft. Individual fairness guarantees for neural networks. In *International Joint Conference on Artificial Intelligence*, 2022.

[9] Marianne Bertrand and Sendhil Mullainathan. Are emily and greg more employable than lakisha and jamal? a field experiment on labor market discrimination. *American Economic Review*, 94(4):991–1013, September 2004.

[10] Sumon Biswas and Hridesh Rajan. Fairify: Fairness verification of neural networks. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 1546–1558, 2023.

[11] Fabian Boemer, Rosario Cammarota, Daniel Demmler, Thomas Schneider, and Hossein Yalame. Mp2ml: A mixed-protocol machine learning framework for private inference. In *Proceedings of the 15th international conference on availability, reliability and security*, pages 1–10, 2020.

[12] Gautam Botrel, Thomas Piellard, Youssef El Housni, Ivo Kubjas, and Arya Tabaie. Consensys/gnark: v0.9.0, February 2023.

[13] Tim Brennan, William Dieterich, and Beate Ehret. Evaluating the predictive validity of the compas risk and needs assessment system. *Criminal Justice and Behavior*, 36(1):21–40, 2009.

[14] Stephen Casper, Carson Ezell, Charlotte Siegmann, Noam Kolt, Taylor Lynn Curtis, Benjamin Bucknall, Andreas Haupt, Kevin Wei, Jérémy Scheurer, Marius Hobbhahn, et al. Black-box access is insufficient for rigorous ai audits. In *The 2024 ACM Conference on Fairness, Accountability, and Transparency*, pages 2254–2272, 2024.

[15] Francesco Croce, Maksym Andriushchenko, and Matthias Hein. Provable robustness of relu networks via maximization of linear regions. In *the 22nd International Conference on Artificial Intelligence and Statistics*, pages 2057–2066. PMLR, 2019.

[16] J Dastin. Amazon scraps secret ai recruiting tool that showed bias against women, October 2018.

[17] Amit Datta, Michael Carl Tschantz, and Anupam Datta. Automated experiments on ad privacy settings: A tale of opacity, choice, and discrimination. *ArXiv*, abs/1408.6491, 2014.

[18] Alice Doherty, Matthew Wicker, Luca Laurenti, and Andrea Patane. Individual fairness in bayesian neural networks, 2023.

[19] 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*, ITCS '12, page 214–226, New York, NY, USA, 2012. Association for Computing Machinery.

[20] Cynthia Dwork and Martha Minow. Distrust of artificial intelligence: Sources & responses from computer science & law. *Daedalus*, 151(2):309–321, 2022.

[21] Boyuan Feng, Lianke Qin, Zhenfei Zhang, Yufei Ding, and Shumo Chu. Zen: Efficient zero-knowledge proofs for neural networks. *IACR Cryptol. ePrint Arch.*, 2021:87, 2021.

[22] Kazuto Fukuchi, Satoshi Hara, and Takanori Maehara. Faking fairness via stealthily biased sampling, 2019.

[23] Sanjam Garg, Aarushi Goel, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, Guru-Vamsi Policharla, and Mingyuan Wang. Experimenting with zero-knowledge proofs of training. Cryptology ePrint Archive, Paper 2023/1345, 2023. `https://eprint.iacr.org/2023/1345`.

[24] Bishwamittra Ghosh, D. Basu, and Kuldeep S. Meel. Justicia: A stochastic sat approach to formally verify fairness. In *AAAI Conference on Artificial Intelligence*, 2020.

[25] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *J. ACM*, 38(3):690–728, jul 1991.

[26] 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.

[27] 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.

[28] Faisal Hamman, Jiahao Chen, and Sanghamitra Dutta. Can querying for bias leak protected attributes? achieving privacy with smooth sensitivity. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, pages 1358–1368, 2023.

[29] Boris Hanin and David Rolnick. Deep relu networks have surprisingly few activation patterns. *Advances in neural information processing systems*, 32, 2019.

[30] Hans Hofmann. Statlog (German Credit Data). UCI Machine Learning Repository, 1994. DOI: https://doi.org/10.24432/C5NC77.

[31] Rashidul Islam, Shimei Pan, and James R Foulds. Can we obtain fairness for free? In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 586–596, 2021.

[32] Philips George John, Deepak Vijaykeerthy, and Diptikalyan Saha. Verifying individual fairness in machine learning models, 2020.

[33] Matt Jordan, Justin Lewis, and Alexandros G. Dimakis. *Provable Certificates for Adversarial Examples: Fitting a Ball in the Union of Polytopes*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[34] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. {GAZELLE}: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1651–1669, 2018.

[35] Daniel Kang, Tatsunori Hashimoto, Ion Stoica, and Yi Sun. Scaling up trustless dnn inference with zero-knowledge proofs, 2022.

[36] Mintong Kang, Linyi Li, Maurice Weber, Yang Liu, Ce Zhang, and Bo Li. Certifying some distributional fairness with subpopulation decomposition, 2022.

[37] 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.

[38] Haitham Khedr and Yasser Shoukry. Certifair: A framework for certified global fairness of neural networks, 2022.

[39] Niki Kilbertus, Adria Gascon, Matt Kusner, Michael Veale, Krishna Gummadi, and Adrian Weller. Blind justice: Fairness with encrypted sensitive attributes. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2630–2639. PMLR, 10–15 Jul 2018.

[40] Seunghwan Lee, Hankyung Ko, Jihye Kim, and Hyunok Oh. vcnn: Verifiable convolutional neural network. *IACR Cryptol. ePrint Arch.*, 2020:584, 2020.

[41] Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 619–631, 2017.

[42] Tianyi Liu, Xiang Xie, and Yupeng Zhang. zkcnn: Zero knowledge proofs for convolutional neural network predictions and accuracy. *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021.

[43] Pranav Maneriker, Codi Burley, and Srinivasan Parthasarathy. Online fairness auditing through iterative refinement. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 1665–1676, New York, NY, USA, 2023. Association for Computing Machinery.

[44] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Comput. Surv.*, 54(6), jul 2021.

[45] Payman Mohassel and Peter Rindal. Aby3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 35–52, 2018.

[46] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*, pages 19–38. IEEE, 2017.

[47] Saerom Park, Seongmin Kim, and Yeon-sup Lim. Fairness audit of machine learning models with confidential computing. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 3488–3499, New York, NY, USA, 2022. Association for Computing Machinery.

[48] Sikha Pentyala, David Melanson, Martine De Cock, and Golnoosh Farnadi. Privfair: a library for privacy-preserving fairness auditing, 2022.

[49] Haakon Robinson, Adil Rasheed, and Omer San. Dissecting deep neural networks. *arXiv preprint arXiv:1910.03879*, 2019.

[50] Anian Ruoss, Mislav Balunovic, Marc Fischer, and Martin Vechev. Learning certified individually fair representations. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7584–7596. Curran Associates, Inc., 2020.

[51] Shahar Segal, Yossi Adi, Benny Pinkas, Carsten Baum, Chaya Ganesh, and Joseph Keshet. Fairness in the eyes of the data: Certifying machine-learning models. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '21, page 926–935, New York, NY, USA, 2021. Association for Computing Machinery.

[52] Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. Bounding and counting linear regions of deep neural networks. In *International Conference on Machine Learning*, pages 4558–4566. PMLR, 2018.

[53] Ali Shahin Shamsabadi, Sierra Calanda Wyllie, Nicholas Franzese, Natalie Dullerud, Sébastien Gambs, Nicolas Papernot, Xiao Wang, and Adrian Weller. Confidential proof of fair training of trees. *ICLR*, 2023.

[54] Ioana Baldini Soares, Chhavi Yadav, Payel Das, and Kush Varshney. Keeping up with the language models: Robustness-bias interplay in nli data and models. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

[55] Wenting Zheng Srinivasan, PMRL Akshayaram, and Popa Raluca Ada. Delphi: A cryptographic inference service for neural networks. In *Proc. 29th USENIX Secur. Symp*, pages 2505–2522, 2019.

[56] Haochen Sun and Hongyang Zhang. zkdl: Efficient zero-knowledge proofs of deep learning training, 2023.

[57] Ehsan Toreini, Maryam Mehrnezhad, and Aad van Moorsel. Verifiable fairness: Privacy-preserving computation of fairness for machine learning systems. 2023.

[58] Caterina Urban, Maria Christakis, Valentin Wüstholz, and Fuyuan Zhang. Perfectly parallel fairness certification of neural networks. *Proc. ACM Program. Lang.*, 4(OOPSLA), nov 2020.

[59] N Vigdor. Apple card investigated after gender discrimination complaints., November, 2019.

[60] Sheridan Wallarchive and Hilke Schellmannarchive. Linkedin's job-matching ai was biased. the company's solution? more ai., June, 2021.

[61] Jiasi Weng, Jian Weng, Gui Tang, Anjia Yang, Ming Li, and Jia-Nan Liu. Pvcnn: Privacy-preserving and verifiable convolutional neural network testing. *Trans. Info. For. Sec.*, 18:2218–2233, mar 2023.

[62] Shaojie Xu, Joel Vaughan, Jie Chen, Aijun Zhang, and Agus Sudjianto. Traversing the local polytopes of relu neural networks: A unified approach for network verification. *arXiv preprint arXiv:2111.08922*, 2021.

[63] Chhavi Yadav, Michal Moshkovitz, and Kamalika Chaudhuri. A learning-theoretic framework for certified auditing with explanations, 2022.

[64] Tom Yan and Chicheng Zhang. Active fairness auditing, 2022.

[65] I-Cheng Yeh. default of credit card clients. UCI Machine Learning Repository, 2016. DOI: https://doi.org/10.24432/C55S3H.

[66] Samuel Yeom and Matt Fredrikson. Individual fairness revisited: Transferring techniques from adversarial robustness. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20, 2021.

[67] Mikhail Yurochkin, Amanda Bower, and Yuekai Sun. Training individually fair ml models with sensitive subspace robustness. In *International Conference on Learning Representations*, 2020.

[68] Jiaheng Zhang, Zhiyong Fang, Yupeng Zhang, and Dawn Song. Zero knowledge proofs for decision tree predictions and accuracy. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 2039–2053, New York, NY, USA, 2020. Association for Computing Machinery.

[69] Lingchen Zhao, Qian Wang, Cong Wang, Qi Li, Chao Shen, and Bo Feng. Veriml: Enabling integrity assurances and fair payments for machine learning as a service. *IEEE Transactions on Parallel and Distributed Systems*, 32(10):2524–2540, 2021.

A Background Cntd.

A.1 Polytopes

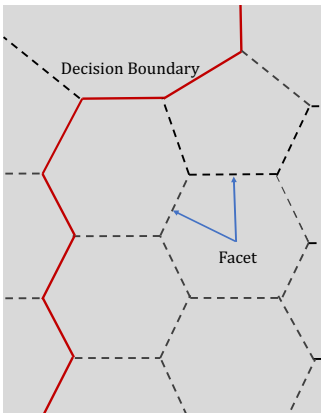


Figure 6: A neural network with ReLU activations partitions the input space into polytopes.

The polytopes described succinctly by their linear inequalities (i.e., they are H-polytopes), which means that the number of halfspaces defining the polytope, denoted by m, is at most $O(poly(n))$, i.e. polynomial in the ambient dimension.

Next, we present a lemma which states that slicing a polyhedral complex with a hyperplane also results in a polyhedral complex.

Lemma A.1. *Given an arbitrary polytope $\mathcal{P} := \{x | Ax \leq B\}$ and a hyperplane $H := \{x | c^T x = d\}$ that intersects the interior of $\mathcal{P}$, the two polytopes formed by the intersection of $\mathcal{P}$ and the each of closed halfspaces defined by H are polyhedral complices.*

Fact A.2. *Two ReLU activation codes of two neighboring polytopes differ in a single position and the differing bit corresponds to the facet common to both.*

B Individual Fairness Certification Cntd.

Algorithm.

In this section, we describe the concrete algorithm to compute the local individiual fairness parameter for a data point x^* (Algorithm 1). Our construction is based on the Geocert algorithm by Jordan et. al (Algorithm 2, Section 2) for computing the pointwise ℓ_2 robustness of neural networks with two key distinctions. First, we run on all the union of $(n-k)$-dimensional polytopes each of which corresponds to a fixed value of the sensitive feature set $\mathcal{S}$. Second, for each of these complices, we compute a lower bound on the pointwise ℓ_2 robustness. The final certificate of fairness is the minimum over all the above bounds.

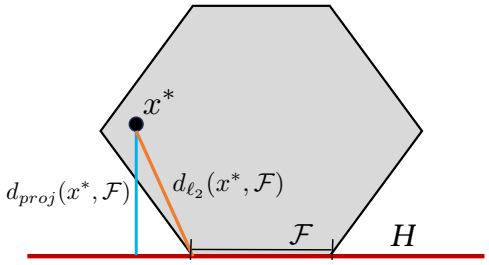


Figure 7: Projection of x^* onto the hyperplane H containing facet $\mathcal{F}$ gives a lower bound on the ℓ_2 distance between x^* and $\mathcal{F}$, i.e., $d_{proj}(x^*, \mathcal{F}) \leq d_{\ell_2}(x^*, \mathcal{F})$.

In the following, we describe the working of the algorithm 1 in more detail. First, we compute the polyhedral complex $\mathbb{P}$ for the model f (Step 1). Next for a fixed value of the set of the sensitive features $\mathcal{S}$ (Step 3), we compute the corresponding $(n-k)$-dimensional polyhedral complex $\mathbb{P}'$ from the original n-dimensional polyhedral complex (ReducePolyDim function Alg. 3). The key idea is to fix the corresponding values of the features in $\mathcal{S}$ in the linear constraints of the polytopes in $\mathbb{P}$. In the next step, we compute a lower bound on the pointwise ℓ_2 robustness of x^* for the polyhedral complex $\mathbb{P}'$ using the Geocert algorithm (Step 5-6). In particular, instead of minimizing the ℓ_2 distance to a facet $\mathcal{F}$, we compute the projection of x^* onto a hyperplane H, where $\mathcal{F}$ lies entirely on H. The above computation is repeated for all the values of the set of sensitive features $\mathcal{S}$. The final certificate of fairness is the minimum of all the lower bounds as computed above (Step 8).

In what follows, we briefly describe how to compute of the pointwise ℓ_2 robustness of a point x. The problem essentially boils down to computing the largest ℓ_2 ball centered at x that fits within the union of n-dimensional polytopes defined by f.

Algorithm 2 Geocert: Pointwise ℓ_2 Robustness

Input x^* - Data point for pointwise ℓ_2 robustness certification; f - Neural network; $dist$ - Distance Metric;
Output ϵ - Pointwise ℓ_2 robustness certificate on x^*;

1: Compute all the polytopes for f
2: Setup priority queue $Q \leftarrow [\,]$
3: Setup list of seen polytopes $C \leftarrow \{\mathcal{P}(x)\}$ ▷ $\mathcal{P}(x)$ denotes the polytope containing x
4: **For** Facet $\mathcal{F} \in \mathcal{P}(x)$ **do**
5: $\quad Q.push(\mathsf{ComputeDistance}(\mathcal{F}, x^*), \mathcal{F}, dist)$
6: **End For**
7: **While** $Q \neq \emptyset$ **do**
8: $\quad (d, \mathcal{F}) \leftarrow Q.pop()$
9: $\quad$ **If** $\mathsf{IsBoundary}(\mathcal{F}) == 1$
10: $\quad\quad$ **Return** d
11: $\quad$ **Else**
12: $\quad\quad$ **For** $\mathcal{P} \in \mathcal{N}(\mathcal{F}) \setminus C$ **do**
▷ $\mathcal{N}(\mathcal{F})$ denote the two polytopes sharing the facet $\mathcal{F}$
13: $\quad\quad\quad$ **For** $\mathcal{F} \in \mathcal{P}$ **do**
14: $\quad\quad\quad\quad Q.push(\mathsf{ComputeDistance}(\mathcal{F}, x^*), \mathcal{F}, dist)$
15: $\quad\quad\quad$ **End For**
16: $\quad\quad$ **End For**
17: $\quad$ **End If**
18: **End While**

Algorithm 3 ReducePolyDim : Construct $(n-k)$-dimensional polytopes from n-dimensional polytopes

Inputs $\mathbb{P} = \bigcup \mathcal{P}$: Set of Polytopes where each polytope $\mathcal{P}$ is expressed as $\{x|\mathbf{A}x \leq \mathbf{b}\}$, $s = (s_1, \cdots, s_k)$: Values of k sensitive features
Output $\mathbb{P}'$: Set of $(n-k)$-dimensional Polytopes

1: $\mathbb{P}' := \{\}$
2: **for** $\mathcal{P} \in \mathbb{P}$
3: $\quad$ **for** $i \in |row(\mathbf{A})|$
4: $\quad\quad$ **for** $j \in [k+1, n]$
5: $\quad\quad\quad \mathbf{A}'[i][j-k] = \mathbf{A}'[i][j]$
6: $\quad\quad$ **end for**
7: $\quad\quad \mathbf{b}'[i] = \mathbf{b}[i] - \sum_{j=1}^{k} \mathbf{A}[i][j] \cdot s_j$
8: $\quad$ **end for**
9: $\quad$ Express $\mathcal{P}' = \{x|\mathbf{A}'x \leq \mathbf{b}'\}$
10: $\quad \mathbb{P}' := \mathbb{P}' \cup \mathcal{P}'$
11: **end for**
12: **Return** $\mathbb{P}'$

Algorithm 4 *FairProof*: Verifiable Individual Fairness Certification

Input x^* - Data point for fairness certification; $\mathbf{W}$ - Weights of the piecewise linear neural network;
Output ϵ - Local individual fairness parameter for x; $\mathsf{com}_{\mathbf{W}}$ - Commitment to the weights of the model; ZK proof that the ϵ is indeed a lower bound on ϵ_{IF}
Pre-Processing Offline Phase
Construct the polyhedral complex $\mathbb{P} = \bigcup \mathcal{P}$ from $\mathbf{W}$ where each polytope is expressed as $\mathcal{P} = \{x | \mathbf{A}x \leq \mathbf{b}\}$
Compute a reference point z_i for each polytope $\mathcal{P}_i \in \mathbb{P}$ such that $z_i \in \mathcal{P}_i$
Commit to the model weights $\mathsf{com}_{\mathbf{W}}$ and release them publicly
Online Phase
$E = [\,]$
for $(s_1, \cdots, s_k) \in domain(S_1) \times \cdots \times domain(S_k)$
  for $\mathcal{P} \in \mathbb{P}$
    for $i \in |row(\mathbf{A})|$
      for $j \in [k+1, n]$
        $\mathbf{A}'[i][j-k] = \mathbf{A}'[i][j]$
      end for
      $\mathbf{b}'[i] = \mathbf{b}[i] - \sum_{j=1}^{k} \mathbf{A}[i][j] \cdot s_j$
    end for
    Express $\mathcal{P}' = \{x | \mathbf{A}'x \leq \mathbf{b}'\}$
    $\mathbb{P}' = \mathbb{P}' \cup \mathcal{P}'$
  end for
  $\big(\epsilon', \mathcal{P}_1, \langle(\mathcal{F}_1, d_1), \cdots, (\mathcal{F}_n, d_n)\rangle\big) = \mathsf{GeoCert}(x^*, \mathbb{P}', d_{proj})$
  ▷ $\mathcal{P}_1$ is the first polytope traversed ▷ $\langle(\mathcal{F}_1, d_1), \cdots, (\mathcal{F}_n, d_n)\rangle$ is the ordered sequence of the visited facets and their corresponding distances
  Prover proves that P_1 is the polytope in $\mathbb{P}'$ containing x^* ▷ Using VerifyPolytope
  Initialize the list of seen facets $T = [\,]$
  for facet $\mathcal{F} \in \mathcal{N}(\mathcal{P}_1)$
    Prover proves that the computation of the distance d from x^* to $\mathcal{F}$ is correct ▷ Using VerifyDistance
    $T.insert\big((\mathcal{F}, d)\big)$;
  end for
  for $i \in [m-1]$
    Prover proves that $\mathcal{F}_i$ is indeed the facet with the smallest distance in T ▷ Using VerifyOrder
    Prover proves that $\mathcal{F}$ is not a boundary facet ▷ Using VerifyBoundary
    for $\mathcal{P} \in \mathcal{N}(\mathcal{F}_i)$
      Prover proves that $\mathcal{P}$ is a neighboring polytope sharing facet $\mathcal{F}$ ▷ Using VerifyNeighbor
      for $\mathcal{F} \in \mathcal{N}(\mathcal{P})$
        Prover proves that the computation of the distance d from x^* to $\mathcal{F}$ is correct ▷ Using VerifyDistance
        $T.insert\big((\mathcal{F}, d)\big)$
      end for
    end for
    $T.remove\big((\mathcal{F}_i, d_i)\big)$
  end for
  Prover proves that $\mathcal{F}_m$ is indeed the facet with the smallest distance in T_2 ▷ Using VerifyOrder
  Prover proves that $\mathcal{F}_m$ is a boundary facet ▷ Using VerifyBoundary
  $E.insert\big(d_m\big)$
end for
Prove that $\epsilon = \min E$ ▷ Using VerifyMin

Algorithm 5 VerifyPolytope

Input x^* - Data point for fairness certification; $\mathsf{com}_{\mathbf{W}}$ - Committed weights of the piecewise linear neural network; $(s_1, \cdots, s_k)$ - Values of the sensitive features;
Output $\mathcal{P}'$ - Polytope corresponding to $\mathbf{W}$ containing x^*; $\mathbf{R}$ - ReLU activation code of x^*; π - ZK proof of the computation;

Evaluate x^* on $\mathsf{com}_{\mathbf{W}}$ to obtain ReLU activation code $\mathbf{R}$
Compute the $n-k$-dimensional polytope $\mathcal{P} = \{x | \mathbf{A}x \leq \mathbf{b}\}$ corresponding to $\mathbf{R}$ on $\mathsf{com}_{\mathbf{W}}$ with $(s_1, \cdots, s_k)$ as the values of the sensitive features
Generate proof π of the above computation
return $(\mathcal{P}, \mathbf{R}, \pi)$

Algorithm 6 VerifyDistance

Input x^* - Data point for fairness certification; $\mathcal{F}$ - Facet;
Output d - Projected distance; π - ZK proof of the computation;

Let $\mathcal{F}$ be represented as $a^T \cdot x = b$
Compute $d = (\left| b - a^T x^*)/||a|| \right|$
Generate proof π of the above computation
return (d, π)

Algorithm 7 VerifyNeighbor

Input $\mathsf{com}_{\mathbf{W}}$ - Weights of the piecewise linear neural network; $\mathcal{F}$ - Facet; $\mathcal{P}$ - Current polytope; $\mathbf{R}$ - ReLU activation code for $\mathcal{P}$; z - Representative point for neighboring polytope; $(s_1, \cdots, s_k)$ - Values of the sensitive features;
Output $\mathcal{P}'$ - Neighboring polytope; $\mathbf{R}'$ - ReLU activation code of $\mathcal{P}'$; π - ZK proof of the computation

$(\mathcal{P}', \mathbf{R}', \pi') \leftarrow \mathsf{VerifyPolytope}(z, \mathsf{com}_{\mathbf{W}}, (s_1, \cdots, s_k))$
▷ Can be performed apriori in a pre-processing stage for efficiency
if $(|\mathbf{R} - \mathbf{R}'|_1 \neq 1)$ ▷ Check hamming distance 1 between two binary vectors
  return $\perp$
if $(\mathcal{F} \notin \mathcal{N}(\mathcal{P}') \wedge (\mathcal{F} \notin \mathcal{N}(\mathcal{P})))$ ▷ Check facet $\mathcal{F}$ is common to both the polytopes
  return $\perp$
Generate proof π of the above computation
return $(\mathcal{P}', \mathbf{R}', (\pi, \pi'))$

Algorithm 8 VerifyBoundary

Input x^* - Data point for fairness certification; $\mathsf{com}_{\mathbf{W}}$ - Weights of the piecewise linear neural network; $\mathcal{F}$ - Current facet represented as $\{x | \mathbf{A}x \leq \mathbf{b}\}$; $\mathcal{P}$ - Current polytope; $\mathbf{R}$ - ReLU activation code for $\mathcal{P}$; z - Representative point for current facet $\mathcal{F}$ $(s_1, \cdots, s_k)$ - Values of the sensitive features;

Output b - Bit indicating boundary condition; π - ZK proof of the computation

Compute the linear function $f_{\mathbf{R}}$ corresponding to activation code $\mathbf{R}$ on $\mathsf{com}_{\mathbf{W}}$ with $(s_1, \cdots, s_k)$ as the values of the sensitive features
▷ Can be performed apriori in a pre-processing stage for efficiency
if $(\mathbf{A}z > \mathbf{b})$
  return $\perp$
end if
$b = 1$
for $i \in [1, |\mathcal{Y}| - 1]$
  $b \leftarrow b \cdot (f_{\mathbf{R}}(z)[0] == f_{\mathbf{R}}(z)[i])$
▷ Testing that $f_{\mathbf{R}}(z)$ is equal on all of its elements
end for
Generate proof π of the above computation
return $(b, (\pi, \pi'))$

Algorithm 9 VerifyOrder

Input $(\mathcal{F}, d)$ - Current facet with distance d; $\mathbf{F} = \{(\mathcal{F}_1, d_1), \cdots, (\mathcal{F}_k, d_k)\}$ - List of all previously unseen facets and their distances;
Output π - ZK proof of the computation

for $\mathcal{F}_i \in \mathbf{F}$
    if $(d > d_i)$
        return $\perp$
    end if
end for
Generate proof π of the above computation
return π

Algorithm 10 VerifyMin

Input E - List of values; ϵ^* - Individual fairness parameter;
Output π - ZK proof of the computation

for $\epsilon \in E$
    if $(\epsilon^* > \epsilon)$
        return $\perp$
    end if
end for
Generate proof π of the above computation
return π

Algorithm 11 VerifyInference

Input x^* - Data point for fairness certification; $\mathsf{com}_{\mathbf{W}}$ - Committed weights of the piecewise linear neural network f;
Output y - The prediction $f(x^*)$; π - ZK proof of the computation;

Evaluate x^* on $\mathsf{com}_{\mathbf{W}}$ to obtain ReLU activation code $\mathbf{R}$
Compute the linear function $f_{\mathbf{R}}$ corresponding to activation code $\mathbf{R}$ on $\mathsf{com}_{\mathbf{W}}$
Compute $f_{\mathbf{R}}(x^*)$
$y = \arg\max_{i \in [|\mathcal{Y}|]} f_{\mathbf{R}}(x^*)$
Generate proof π of the above computation
return (y, π)

C Correctness of *FairProof*

In this section, we prove the correctness of *FairProof* given in Alg. 4. First, we re-state the correctness of GeoCert.

Theorem C.1 (Correctness of GeoCert [33])**.** *For a fixed polyhedral complex $\mathbb{P}$, a fixed point x^* and a distance function ϕ that satisfies ray monotonocity, GeoCert returns a boundary facet with the minimum distance.*

Fact C.2. *The projection of a given point x^* onto a hyperplane H where $\mathcal{F} \subseteq H$ gives a lower bound on its ℓ_2 distance to $\mathcal{F}$, i.e., $d_{proj}(x, \mathcal{F}) \leq d_{\ell_2}(x, \mathcal{F})$.*

Theorem C.3. *Let f be a piecewise-linear neural network. Replacing in Algorithm 2 with $d_{\ell_2}(\cdot)$ distance with $d_{proj}(\cdot)$ gives a lower bound on the individual fairness guarantee, i.e., $\epsilon_{d_{proj}} \leq \epsilon_{d_{\ell_2}}$.*

Proof. We will prove by contradiction. Let $\mathbb{P}$ be the polyhedral complex associated with the model f. Let us assume that there exists a boundary facet $\mathcal{F}^*$ such that $d_{\ell_2}(\mathcal{F}, x) < \epsilon_{d_{proj}}$. Now if the corresponding polytope $\mathcal{P}_{\mathcal{F}^*}$ was traversed by $\mathsf{GeoCert}(x, \mathbb{P}, d_{proj})$, then all the facets in $\mathcal{P}_{\mathcal{F}^*}$ including $\mathcal{F}^*$ were checked. Then from the correctness of GeoCert (Thm. C.1), this leads to a contradiction of C.2. Now let us consider the alternative case where $\mathcal{P}_{\mathcal{F}^*}$ was not traversed by $\mathsf{GeoCert}(x, \mathbb{P}, d_{proj})$. From Thm. C.1 this means that there exists another boundary facet $\mathcal{F}^*$ such that $d_{proj}(x, \mathcal{F}^*) \leq d_{proj}(x, \mathcal{F})$. Then by Fact C.2, $d_{proj}(\mathcal{F}^*, x) = \epsilon_{d_{proj}} \leq d_{proj}(\mathcal{F}, x) \leq d_{\ell_2}(\mathcal{F}, x)$ which contradicts our assumption. □

Theorem C.4 (Correctness of *FairProof*)**.** *For a given data point x^*, FairProof (Algorithm 4) generates ϵ such that $\epsilon \leq \epsilon_{IF}$.*

Proof. The proof of the above theorem follows directly from Theorem C.1, Theorem C.3 and Fact C.2. □

D Security Proof

1. **Completeness**

$$\forall x, \mathbf{W} \tag{4}$$

$$\Pr\begin{bmatrix} \mathsf{pp} \leftarrow \textit{FairProof}.\mathsf{KeyGen}(1^\lambda) \\ \mathsf{com}_{\mathbf{W}} \leftarrow \textit{FairProof}.\mathsf{Commit}(\mathbf{W}, \mathsf{pp}, r) \\ (y, \epsilon, \pi) \leftarrow \textit{FairProof}.\mathsf{Prove}(\mathbf{W}, x, \mathsf{pp}, r) \\ \textit{FairProof}.\mathsf{Verify}(\mathsf{com}_{\mathbf{W}}, x, y, \epsilon, \pi, \mathsf{pp}) = 1 \end{bmatrix} = 1 \tag{5}$$

2. **Soundness**

$$\Pr\begin{bmatrix} \mathsf{pp} \leftarrow \textit{FairProof}.\mathsf{KeyGen}(1^\lambda) \\ (\mathbf{W}^*, \mathsf{com}_{\mathbf{W}^*}, \mathbf{X}, \epsilon^*, y^*, \pi^*, r) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}) \\ \mathsf{com}_{\mathbf{W}^*} \leftarrow \textit{FairProof}.\mathsf{Commit}(\mathbf{W}^*, r)) \\ \textit{FairProof}.\mathsf{Verify}(\mathsf{com}_{\mathbf{W}^*}, x, y^*, \epsilon^*, \pi^*, \mathsf{pp}) = 1 \\ \big(\exists \tilde{x}, d(x, \tilde{x}) \leq \epsilon \wedge f(\mathbf{W}^*, \mathbf{X}) \neq f(\mathbf{W}^*, \tilde{\mathbf{X}})\big) \\ \qquad\qquad \vee y \neq f(\mathbf{W}^*, \mathbf{X}) \end{bmatrix} < negl(\lambda) \tag{6}$$

3. **Zero-Knowledge** Let λ be the security parameter obtained from $\lambda, \mathsf{pp} \leftarrow \textit{FairProof}.KeyGen(1^\lambda)$

$$|\Pr[\mathsf{Real}_{\mathcal{A},\mathbf{W}}(\mathsf{pp}) = 1] - \Pr[\mathsf{Ideal}_{\mathcal{A},\mathcal{S}^{\mathcal{A}}}(\mathsf{pp}) = 1]| \leq \mathsf{negl}(\lambda) \tag{7}$$

Proof Sketch. **Completeness.** The completeness guarantee follows trivially from our construction.

Soundness. $\mathcal{L}(x)$ denotes the leakage function for *FairProof*, specifically, $\mathcal{L}(x) = \{n_1, \cdots, n_{|\mathcal{S}|}\}$, where n_i denotes the number of facets traversed for the i-th value of the sensitive attribute $\mathcal{S}$.

Recall, the functioning of GeoCert can be summarized as follows:

$\mathsf{Real}_{\mathcal{A},\mathbf{W}}(\mathsf{pp})$:

1. $\mathsf{com}_{\mathbf{W}} \leftarrow$ *FairProof*.$\mathsf{Commit}(\mathbf{W}, \mathsf{pp}, r)$
2. $x \leftarrow \mathcal{A}(\mathsf{com}_{\mathbf{W}}, \mathsf{pp})$
3. $(y, \epsilon, \pi) \leftarrow$ *FairProof*.$\mathsf{Prove}(\mathbf{W}, x, \mathsf{pp}, r)$
4. $b \leftarrow \mathcal{A}(\mathsf{com}_{\mathbf{W}}, x, y, \epsilon, \pi, \mathsf{pp})$
5. Output b

$\mathsf{Ideal}_{\mathcal{A},\mathcal{S}^{\mathcal{A}}}(\mathsf{pp}, h)$:

1. $\mathsf{com} \leftarrow \mathcal{S}_1(1^\lambda, \mathsf{pp}, r)$
2. $x \leftarrow \mathcal{A}(\mathsf{com}, \mathsf{pp})$
3. $(y, L_x, \epsilon, \pi) \leftarrow \mathcal{S}_2^{\mathcal{A}}(\mathsf{com}, x, \mathsf{pp}, r)$ given oracle access to $y = \mathsf{pred}(\mathbf{W}, x)$, $L_x = \mathcal{L}(x)$ and $\epsilon = IF_{LB}(\mathbf{W}, x)$
4. $b \leftarrow \mathcal{A}(\mathsf{com}_{\mathbf{W}}, x, y, L_x, \epsilon, \pi, \mathsf{pp})$
5. Output b

Figure 8: Zero-knowledge games

1. Start traversing from the polytope containing x^*.
2. Compute the distances to all the facets of the current polytope and store them.
3. Select the hitherto unseen facet with the smallest distance.
4. Stop if this is a boundary facet.
5. Else, traverse next to the neighboring polytope that shares the current facet.

A malicious prover can cheat in any (or a combination) of the above steps. We will consider each of them separately as follows.

Lemma D.1 (Soundness of VerifyPolytope)**.** *Let $\mathcal{P} = \{x | \mathbf{A}x \leq \mathbf{B}\}$ be the correct polytope obtained from the piecewise-linear neural network with weights $\mathbf{W}$ for a given value of the sensitive features. For any polytope $\mathcal{P}' = \{\mathbf{A}'x < \mathbf{b}'\}$ such that $(\mathbf{A} \neq \mathbf{A}') \vee (\mathbf{b} \neq \mathbf{b}')$, we have*

$$\Pr[\textit{FairProof}.\mathsf{Verify}(\mathsf{com}_{\mathbf{W}^*}, x, y^*, \epsilon^*, \pi^*, \mathsf{pp}) = 1] < \mathsf{negl}(\lambda) \tag{8}$$

Proof Sketch. As shown in Alg. 5, the verification process re-computes the correct polytope from the committed model weights. The only way the prover can cheat is if they can produce a P' such that $Open(\mathsf{com}_{\mathcal{P}}) = \mathcal{P}'$ which violates the binding property of the commitment scheme. □

Lemma D.2 (Soundness of VerifyDistance)**.** *For a given facet $\mathcal{F} = \{Ax \leq b\}$, data point x^*, and value d' such that $d' \neq \left|\frac{b - A^T x^*}{\|A\|}\right|$, we have:*

$$\Pr[\textit{FairProof}.\mathsf{Verify}(\mathsf{com}_{\mathbf{W}^*}, x, y^*, \epsilon^*, \pi^*, \mathsf{pp}) = 1] < \mathsf{negl}(\lambda) \tag{9}$$

Proof Sketch. The verification process (Alg. 6) re-computes the correct distance. Hence, the only way the prover can cheat is if they can produce a d' such that $Open(\mathsf{com}_d) = d'$ which violates the binding property of the commitment scheme. □

Lemma D.3 (Soundness of VerifyOrder)**.** *Let $\mathbf{d} = \{d_1, \cdots, d_k\}$ be a set of values such that $d_{min} = \min_i d_i$. For any value d' such that $d' > d_{min}$, we have:*

$$\Pr[\textit{FairProof}.\mathsf{Verify}(\mathsf{com}_{\mathbf{W}}, x, y^*, \epsilon^*, \pi^*, \mathsf{pp}) = 1] < \mathsf{negl}(\lambda) \tag{10}$$

Proof Sketch. The verification checks the minimality of the given value against all values in **d** (Alg. 9). The only way to cheat would require producing a **d** with a different minimum which violates the binding property of the commitment scheme. □

Lemma D.4 (Soundness of VerifyBoundary)**.** *Consider a piecewise-linear neural network with weights $\mathbf{W}$. For any facet $\mathcal{F}$ such that which is not a boundary facet, we have*

$$\Pr[\textit{FairProof}.\mathsf{Verify}(\mathsf{com}_{\mathbf{W}}, x, y^*, \epsilon^*, \pi^*, \mathsf{pp}) = 1] \leq \mathsf{negl}(\lambda) \tag{11}$$

Proof Sketch. The verification algorithm computes the linear function corresponding to the given activation code (Alg. 8. A prover can cheat here only if they can compute a different linear function f' which would require violating the binding property of the commitment scheme. □

Lemma D.5 (Soundness of VerifyNeighbor)**.** *Let $\mathcal{P} = \{x|\mathbf{A}x \leq \mathbf{b}\}$ be a polytope belonging to the polyhedral complex of the piecewise-linear neural network with weights $\mathbf{W}$ and let $\mathcal{F} \in \mathcal{N}(\mathcal{P})$. Let $\bar{\mathcal{P}} = \{x|\bar{\mathbf{A}}x \leq \bar{\mathbf{b}}\}$ and $\mathcal{P}$ be neighboring polytopes, sharing the facet $\mathcal{F}$, i.e., $\bar{\mathcal{P}} \in \mathcal{N}(\mathcal{F}) \setminus \mathcal{P}$. Let $z \in \mathbf{R}^n$ be a data point. For any polytope $P' = \{x|\mathbf{A}'x \leq \mathbf{b}'\}$ such that $(\bar{\mathbf{A}} \neq \mathbf{A}') \wedge (\bar{\mathbf{b}} \neq \mathbf{b}')$, we have*

$$\Pr[\textit{FairProof}.\mathsf{Verify}(\mathsf{com}_{\mathbf{W}}, x, y^*, \epsilon^*, \pi^*, \mathsf{pp}) = 1] < \mathsf{negl}(\lambda) \tag{12}$$

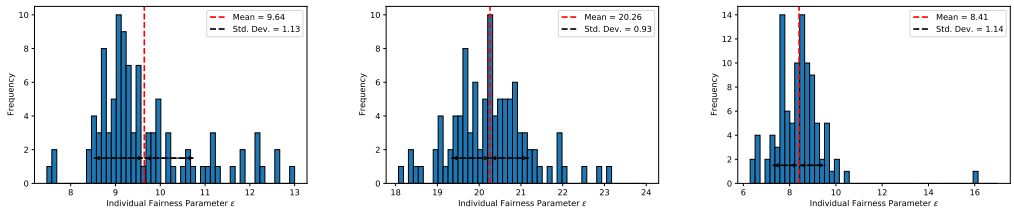
Proof Sketch. The verification algorithm first checks whether $\bar{P}$ contains the reference point z (Alg. 7). The soundness of this follows from VerifyPolytope. Cheating on the next steps (checking the hamming distance and facet intersection) means that the prover is essentially able to generate a polytope P' such that $Open(\mathsf{com}_{\mathcal{P}}) = \mathcal{P}'$ which violates the binding property of the commitment scheme. □

Zero-Knowledge. The zero-knowledge property follows directly from the commitment scheme and the zero-knowledge backend proof system we use. We note that the zero-knowledge proof protocol itself is not the focus of this paper; instead, we show how we can use existing zero-knowledge proof protocols to provide verifiable individual fairness certification in a smart way for high efficiency. □

E Evaluation Cntd.

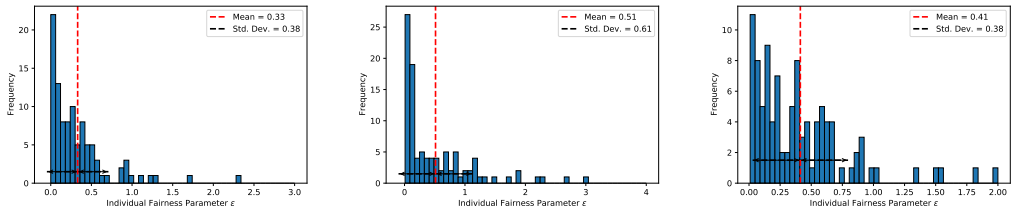
Dataset-Model	Online (in mins)	Offline (in mins)	Improvement	Traversals
German (4,2)	4.90 ± 0.12	3.61 ± 0.19	$1.74\times$	40 ± 3
German (2,4)	1.17 ± 0.02	0.67 ± 0.03	$1.57\times$	13 ± 1
Credit (4,2)	3.52 ± 0.08	2.31 ± 0.10	$1.66\times$	28 ± 2
Credit (2,4)	2.08 ± 0.04	1.11 ± 0.07	$1.49\times$	25 ± 1
Adult (4,2)	3.94 ± 0.10	1.72 ± 0.08	$1.43\times$	41 ± 3
Adult (8,2)	3.94 ± 0.30	1.34 ± 0.08	$1.36\times$	38 ± 8

Table 1: Time for proof generation averaged over 100 randomly sampled data points. Mean and standard error are reported for each dataset-model. Offline computations are done in the initial setup phase of *FairProof* while Online computations are done for every new query. Improvement = (Online time + Offline time)/ Online time. Traversals gives the total number of iterations (also total number of popped facets) of GeoCert ran by *FairProof*.



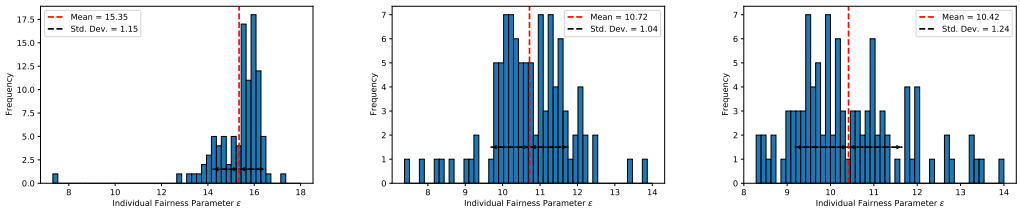
(a) *Credit, wd=2.5* (b) *Adult, wd=0.2* (c) *German, wd=10*

Figure 9: Histogram of fairness parameter ϵ for fair models of size (4,2). 'wd' represents the values of the Weight decay parameter.



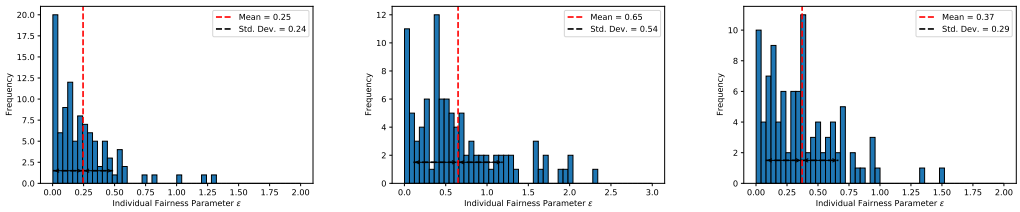
(a) *Credit* (b) *Adult* (c) *German*

Figure 10: Histogram of fairness parameter ϵ for unfair models of size (4,2). Weight decay is set to zero here for all.



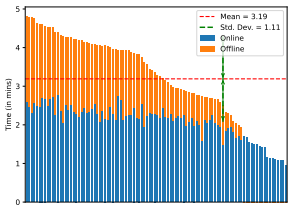
(a) *Credit, wd=3* (b) *Adult, wd=0.2* (c) *German, wd=10*

Figure 11: Histogram of fairness parameter ϵ for fair models of size (8,2). 'wd' represents the values of the Weight decay parameter.

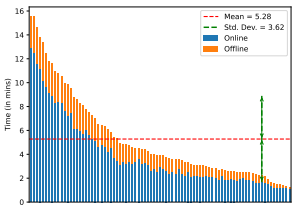


(a) *Credit* (b) *Adult* (c) *German*

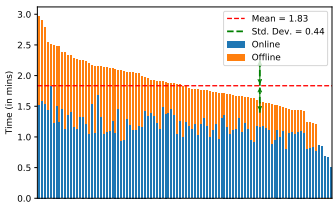
Figure 12: Histogram of fairness parameter ϵ for unfair models of size (8,2). Weight decay is set to zero here for all.



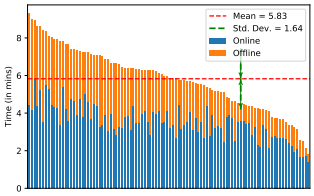
(a) *Credit (2,4)*



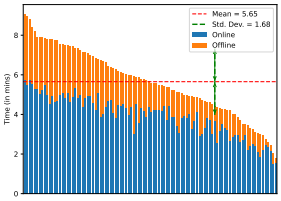
(b) *Adult (8,2)*



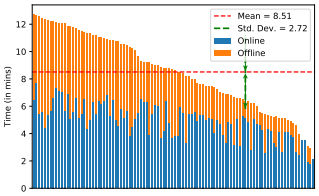
(c) *German (2,4)*



(d) *Credit (4,2)*

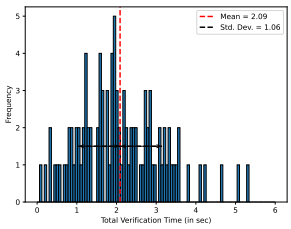


(e) *Adult (4,2)*

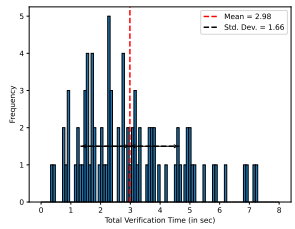


(f) *German (4,2)*

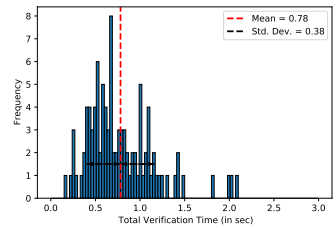
Figure 13: Proof generation time for 100 random data points.



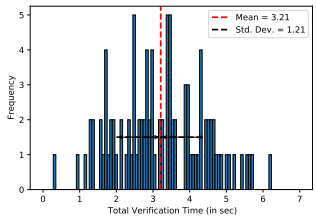
(a) *Credit (2,4)*



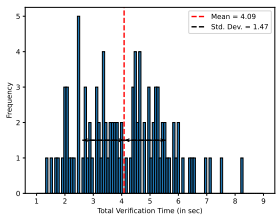
(b) *Adult (8,2)*



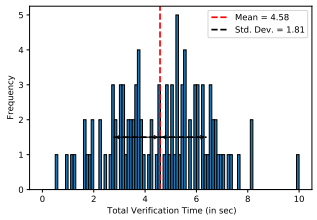
(c) *German (2,4)*



(d) *Credit (4,2)*

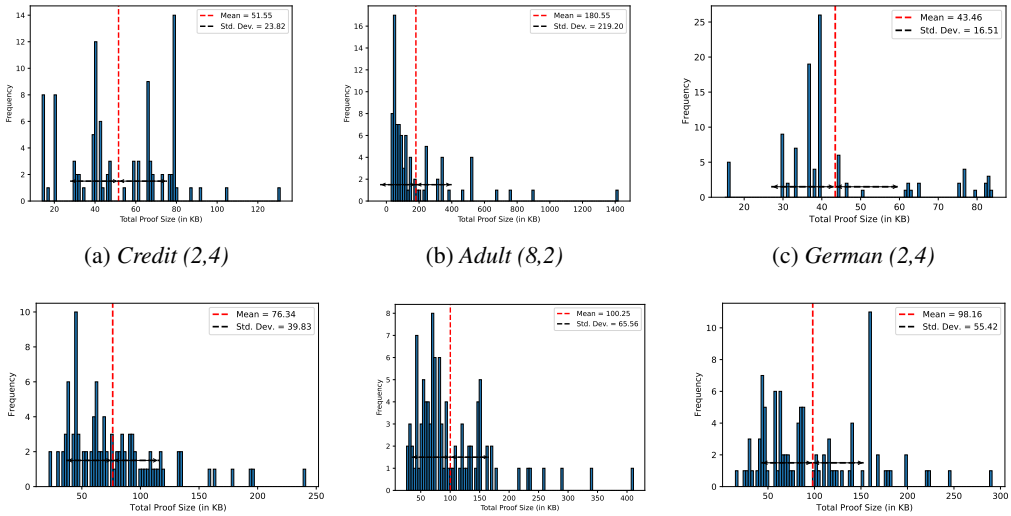


(e) *Adult (4,2)*



(f) *German (4,2)*

Figure 14: Distribution of verification time for 100 random data points.



(a) *Credit (2,4)* (b) *Adult (8,2)* (c) *German (2,4)*

(d) *Credit (4,2)* (e) *Adult (4,2)* (f) *German (4,2)*

Figure 15: Distribution of communication cost (proof size) for 100 random data points.

F Related Work

Certifiable fairness. Prior research on certifying fairness of a ML model can be classified into three types. The first line of work issues a certificate of fairness directly from the model weights by framing it as an optimization problem. [32] presented optimization based mechanisms for certifying the (global) individual fairness of linear classifiers and kernelized classifiers with polynomial/rbf kernels. [8] extended the results to neural networks by encoding (global) individual fairness certification as a mixed-integer linear programming problem. [36] proposed a notion of distributional fairness and give a framework to compute provable certificates for the same.

The second line of research has leveraged the connection between robustness and fairness, and proposed fairness-aware training mechanisms akin to adversarial training. [50] deviced a mechanism for training individually fair representations which can be used to obtain a certificate of individual fairness for the end-to-end model by proving local robustness. SenSR [67] introduced a distributionally robust optimization approach to enforce individual fairness on a model during training. CertiFair [38] enabled certification of (global) individual fairness using off-the-shelf neural network verifiers. Additionally, the authors proposed a fairness aware training methodology with a modified reguralizer. [66] applied randomized smoothing from adversarial robustness to make neural networks individually fair under a given weighted ℓ_p metric. [18] estimated the (global) individual fairness parameter for Bayesian neural networks by designing Fair-FGSM and Fair-PGD – fairness-aware extensions to gradient-based adversarial attacks for BNNs.

The final line of work is based on learning theoretic approaches [63, 64, 43] where a third-party audits the fairness of a model in a query-efficient manner.

The problem of fairness certification has also garnered attention from the formal verification community. FairSquare [3] encoded a range of global fairness definitions as probabilistic program properties and provides a tool for automatically certifying that a program meets a given fairness property. VeriFair [6] used adaptive concentration inequalities to design a probabilistically sound global fairness certification mechanism for neural networks. [58] proposes a static analysis framework for certifying fairness of feed-forward neural networks. Justicia [24] presents a stochastic satisfiability framework for formally verifying different group fairness measures, such as disparate impact, statistical parity, and equalized odds, of supervised learning algorithms. A recent work, Fairify [10], generates a certificate for the global individual fairness of a pre-trained neural network using SMT solvers. It is important to note that all the aforementioned approaches focus on certification in the plain text, i.e., they do not preserve model confidentiality.

Verifiable machine learning. A growing line of work has been using cryptographic primitives to verify certain properties of a ML model without violating its confidentiality. Prior research has primarily focused on verifying the inference and accuracy of models. For instance, [68] proposed a zero-knowledge protocol for tailored for verifying decision trees, while zkCNN [42] introduced an interactive protocol for verifying model inference for convolutional neural networks. Several other works have focused on non-interactive zero-knowledge inference for neural networks, including [61, 2, 35, 56, 21, 40]. VeriML [69] enabled the verification of the training process of a model that has been outsourced to an untrusted third party. [23] proposed a mechanism for generating a cryptographic proof-of-training for logistic regression.

Most of the prior work on verifying fairness while maintaining model confidentiality [48, 39, 57, 51, 47] has approached the problem in the third-party auditor setting. A recent work [53] proposed a fairness-aware training pipeline for decision trees that allows the model owner to cryptographically prove that the learning algorithm used to train the model was fair by design. In contrast, *FairProof* allows a model owner to issue a certificate of fairness of neural networks by simply inspecting the model weights post-training.