Zero-Knowledge Proofs for Gradient Boosted Decision Trees*

Jiacheng Gao[1], Wenjie Qu[2], Yuan Zhang[1] ✉, Sheng Zhong[1], and Jiaheng Zhang[2]

[1] State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing, China
jcgao@smail.nju.edu.cn, zhangyuan@nju.edu.cn, zhongsheng@nju.edu.cn
[2] School of Computing, National University of Singapore, Singapore
wenjiequ@u.nus.edu, jhzhang@nus.edu.sg

Abstract. Gradient boosted decision trees (GBDTs) are widely used for tabular data in finance, healthcare, and risk assessment. As GBDT training and inference are increasingly outsourced, clients need to verify that models are trained as claimed and predictions are consistent with those models, while providers may need to keep training data and model parameters private. This makes zero-knowledge proofs for GBDT training and inference a natural solution.

Existing approaches typically compile training certification into circuits and prove their execution using generic ZKP backends, incurring high prover costs. Directly proving the underlying algebraic relations offers an alternative, but proving each relation separately incurs substantial costs for committing to and opening auxiliary witnesses. Batching is challenging because these witnesses have different shapes and sizes and are encoded over different domains.

We present TERRAE, a zero-knowledge proof system for quantized GBDT training and inference based on KZG polynomial commitments. TERRAE avoids both proving circuit computation and proving each constraint separately by leveraging the structure of GBDT training and two new batching techniques: domain-lifting batching for low-degree algebraic constraints and interleaving batching for same-form non-native constraints. Both techniques work over differently-sized domains and reduce many constraints to a single claim without introducing extra polynomial commitments. We also design a histogram proof that proves the correctness of aggregating sample-wise updates into histograms, which may be of independent interest. Our evaluation shows that, compared with prior work CertXGB instantiated with the same KZG backend, TERRAE achieves proof-generation speedups of 3.18–13.55× for training and 1.96–4.31× for inference across completed benchmark comparisons, with lower peak prover memory, at the cost of moderately larger proofs and higher verification time.

Keywords: Gradient Boosted Decision Trees · Zero-Knowledge Proofs

* This is the full version of [16] which is accepted at ACM CCS 2026.

1 Introduction

Machine-learning models are increasingly trained and deployed through outsourced and cloud-based pipelines. Machine learning as a service (MLaaS) allows clients with limited computational resources or machine-learning expertise to outsource training and inference tasks to external providers [30, 37, 42, 44]. This paradigm is attractive in settings such as risk scoring, healthcare analytics, fraud detection, and financial decision making, where predictive models are often trained or served over sensitive tabular data [8, 18, 35]. However, outsourcing also creates an integrity problem: a client may receive a trained model or a prediction from an external provider, but cannot directly tell whether it was produced by the claimed training pipeline on the claimed data source. Prior work on verifiable MLaaS and proof-of-learning has therefore emphasized the need to certify that outsourced training or inference was executed correctly, rather than replaced by a shortcut, a partially trained model, or an improperly executed computation [1, 20, 44].

At the same time, full transparency is often unacceptable. Training data may contain sensitive user, medical, or financial records, while the learned model itself may encode proprietary information or leak information about the training data [38, 42]. Model extraction attacks further show that even black-box prediction APIs can compromise model confidentiality, especially for models offered through cloud prediction services [38]. This creates a tension between verifiability and confidentiality: the verifier wants assurance that the model was trained correctly, whereas the prover may need to keep the data, model parameters, and intermediate training states private. Zero-knowledge proofs of training address this tension by allowing a prover to convince a verifier that a committed model is the output of a public training algorithm on a committed dataset, while revealing nothing beyond the validity of this statement [1, 28].

We study proofs of training and inference for gradient boosted decision trees (GBDTs). GBDTs, and especially implementations such as XGBoost [8], are among the strongest model families for tabular data and are widely used in applications such as clinical risk prediction, financial risk assessment, and fraud detection. These applications are typically driven by medium-sized tabular datasets with heterogeneous numeric and categorical features, missing values, and irregular decision boundaries, a regime in which tree-based models remain especially strong and often outperform or rival deep tabular models in practice [17, 18, 35]. In such domains, model outputs can affect access to treatment, credit, and other consequential opportunities, so it is important not only that the model be accurate, but also that its training and predictions be auditable, verifiable, and contestable [23, 32].

A GBDT model is an additive ensemble of regression trees: each round trains new trees to fit the residual errors of the current model, and the final prediction is obtained by aggregating the outputs of all trees. This sequential structure makes training certification substantially harder than inference certification. To prove inference, it is enough to show that each input follows a valid path in each committed tree and that the returned scores are aggregated correctly. To prove

training, the prover must also show that every split, every leaf value, and every boosting update is consistent with the committed dataset and the public training rule.

Prior zero-knowledge systems for trees mainly prove inference, accuracy, or properties of trained models. zkDT [41] proves private decision-tree predictions and accuracy, CQ++ [7] improves committed-tree inference, and Confidential-PROFITT [33] certifies training fairness. These works provide important tools for tree-model verification, but they do not give a specialized proof system for the full training semantics of gradient boosting.

The closest line of work is zero-knowledge proof of training for tree ensembles. Sparrow [27] gives a space-efficient zkSNARK for data-parallel circuits and applies it to random-forest training and prediction. Its key idea is to avoid re-running training inside the proof; instead, the prover proves that a certification algorithm accepts the committed forest and dataset. ZKBoost [25] extends this certification approach to XGBoost. It defines a deterministic fixed-point version of XGBoost, builds a certification algorithm for the resulting trainer, and then proves this certifier using a general-purpose ZKP backend. These works demonstrate that proofs of training for tree ensembles are possible, but their proof systems still inherit the cost of proving the computation of large certification circuits.

A natural alternative is to avoid proving the certification computation as a generic circuit and instead prove the algebraic relations that characterize GBDT training directly. However, this approach creates a different bottleneck. GBDT training contains many repeated constraints of two different kinds. The first kind consists of *low-degree algebraic constraints*, which are relations that can be written as low-degree polynomial identities over the field, such as score updates, prefix sums, booleanity checks, and zero checks. The second kind consists of *non-native predicates*: checks whose intended semantics depend on structure outside the field algebra, such as table membership, order, range, integer divisibility, rounding, and maximum selection. These constraints also do not naturally live on one uniform domain, since their inputs have different sizes. Different tensors have different shapes, and their polynomial encodings are committed over different subgroups. Proving each relation separately would introduce a large number of auxiliary commitments and openings, while ordinary same-domain batching does not apply directly.

We present Terrae, a zero-knowledge proof system for GBDT training and inference based on KZG polynomial commitments. Terrae proves that a committed forest is obtained by executing a public GBDT trainer on a committed dataset. The proof of Terrae follows the structure of histogram-based tree training verification [25, 27]: it proves that samples in the dataset are routed to the correct leaves, aggregates them into histograms, proves that all internal-node splits are optimal and all stopping conditions hold, and finally proves the updates of pseudo-residuals are correct. However, unlike previous approaches that compile these certification procedures into large circuits, Terrae represents the training trace as committed tensors and proves the relation among them directly.

Besides handling both low-degree algebraic constraints and non-native predicates, the main technical challenge is batching. Once the training semantics are decomposed into algebraic constraints, the prover faces many similar constraints over different domains. TERRAE introduces two batching tools for these claims. First, *domain-lifting batching* combines low-degree algebraic constraints, such as zero-check and sum-check claims, across different domains. It abstracts a technique in the lookup argument of CQ++ [7], and lifts all polynomials to a common domain and then applies a single random linear combination. Second, *interleaving batching* combines same-form non-native predicates, including lookup-style checks, by virtually interleaving committed vectors of different lengths. Any evaluation of the resulting polynomial can be derived efficiently from existing polynomial commitments and public sparse polynomials, and the batching does not introduce any extra auxiliary witnesses.

We also design a dedicated histogram proof for GBDT training. Prior certification-based approaches treat histogram construction as a RAM-like update procedure [26, 27][3], where each sample updates the memory slot indexed by its leaf, feature, and bin, by reading the value from and writing the updated value back to the same location. In contrast, TERRAE exploits the fact that histogram updates are fixed additive updates with a much simpler access pattern than a generic RAM. It proves the final histogram by binding each bucket to a power of a random challenge and checking a weighted-sum identity between the committed histogram and the sample-wise update stream. This turns histogram verification from a sequence of authenticated memory transitions into a single randomized weighted sum identity. It avoids proving a general RAM computation and gives a proof primitive tailored to the histogram structure that appears throughout tree training.

Our main contributions are as follows:

- We present TERRAE, a proof system that proves the training and inference of GBDTs in zero knowledge. TERRAE avoids compiling each training or inference certification procedure into circuits, and instead proves the algebraic relations directly. TERRAE can also be extended to random forests and XGBoost-style training with minor modifications.
- We introduce domain-lifting batching for low-degree algebraic constraints and interleaving batching for non-native predicates. These techniques batch constraints that arise over differently sized domains and reduce many commitment-opening tasks to a small number of batched claims, without introducing extra witnesses.
- We propose a histogram proof that proves the correctness of converting sample-wise training data and pseudo-residuals into per-node, per-feature, and per-bin histograms. The proof exploits the additive structure of histogram construction and avoids general RAM consistency checking.

[3] This discussion refers to ZKBoost v2. After our original submission, the ZKBoost preprint was updated to v3 on May 12, 2026 [25]. Version 3 replaces its RAM-based histogram proof with weighted-LogUp, which is faster than our histogram argument in the microbenchmark. See Section 6.4.

– We implement TERRAE and evaluate it against CertXGB [25] instantiated with the same KZG backend. The measured proof-generation time is 3.18–13.55× lower for training and 1.96–4.31× lower for inference; peak prover memory is consistently lower, at the cost of moderately larger proofs and greater verifier work.

Sections 2 to 5 present the background, security statement, technical ideas, and protocol; Sections 6 and 7 give the evaluation and conclusion. The appendices contain related work, formal constructions and extensions, the forest semantics (Appendix H), and the complete knowledge-soundness analysis (Appendix I).

2 Preliminaries

2.1 Notation and Polynomial Encoding

We work over a prime field $\mathbb{F} = \mathbb{F}_p$. All vectors, matrices, and tensors are flattened in row-major order before commitment. For $n \in \mathbb{N}$, we write $[n] := \{0, 1, \ldots, n-1\}$. For integers $a \le b$, we write $[a, b) := \{z \in \mathbb{Z} : a \le z < b\}$.

All committed lengths are assumed to be powers of two. To make this precise, we fix an integer $L \ge 1$ such that every committed object is smaller than 2^L, and is padded to a length dividing 2^L, where $2^L \mid (p-1)$. We also fix a primitive 2^L-th root of unity $\omega \in \mathbb{F}$. For every power of two n with $n \mid 2^L$, we define

$$\omega_n := \omega^{2^L/n} \qquad \text{and} \qquad \mathbb{H}_n := \langle \omega_n \rangle.$$

Thus $\mathbb{H}_n$ is the multiplicative subgroup of size n used as the evaluation domain for length-n vectors. Its vanishing polynomial is

$$\nu_n(X) := \prod_{a \in \mathbb{H}_n} (X - a) = X^n - 1.$$

Because all domains are derived from the same root ω, we have $\omega_{nm}^m = \omega_n$ whenever n, m, and nm are powers of two dividing 2^L. This compatibility is used later when we align objects of different shapes.

Let $\lambda_n^{(i)}(X)$ denote the i-th Lagrange basis polynomial on $\mathbb{H}_n$.

For a vector $\mathbf{v} \in \mathbb{F}^n$, its polynomial encoding over $\mathbb{H}_n$ is

$$v(X) := \sum_{i \in [n]} \mathbf{v}[i] \lambda_n^{(i)}(X),$$

where $\mathbf{v}[i]$ denotes the i-th entry of $\mathbf{v}$.

Throughout the paper, we need to manipulate the shapes of a high-dimensional tensor. Given $\mathbf{V} \in \mathbb{F}^{a \times b \times c}$, we write $\mathbf{V}_{\{abc\}} \in \mathbb{F}^{abc}$ for its flattened form, and $\mathbf{V}_{\{a \times bc\}} \in \mathbb{F}^{a \times bc}$ for its flattened form with the first dimension preserved. As a sanity check, note that

$$\mathbf{V}_{\{a \times b \times c\}}[i, j, k] = \mathbf{V}_{\{a \times bc\}}[i, j \cdot c + k] = \mathbf{V}_{\{abc\}}[i \cdot bc + j \cdot c + k].$$

Table 1. Main scalar parameters used throughout the paper.

Symbol	Meaning
$\mathbb{F} = \mathbb{F}_p$	Prime field over which the proof is defined.
L	Global logarithmic domain bound; committed objects are padded to lengths dividing 2^L.
T	Number of boosting rounds.
K	Number of classes; each round trains one tree per class.
M	Number of training samples.
d	Number of features.
B	Number of quantization bins; feature values lie in $[B]$.
q	Fractional precision; the fixed-point scale is 2^q.
μ, μ_q	Public learning rate and its encoding $\mu_q = \lfloor \mu 2^q \rfloor$.
h	Maximum root-to-leaf depth of the trees in the forest.
τ_{split}	Minimum RSS reduction required to split a node.
b	Number of public inference samples in one batch.
N_{tot}	Total number of nodes in the encoded forest, including the dummy node.
$N_{\text{int}}, N_{\text{leaf}}$	Number of internal nodes and leaf nodes in the forest.
$\mathcal{B}$	Public bound for signed values and nonnegative range checks.

A similar notation applies to tensors of other dimensions. For tensors of the same shape, we write $\mathbf{V} + \mathbf{W}$ for their entrywise sum, and $\mathbf{V} \odot \mathbf{W}$ for their Hadamard product. We denote the size of tensor $\mathbf{V}$ by $|\mathbf{V}|$.

2.2 KZG Polynomial Commitments

We use the KZG polynomial commitment scheme (PCS) for univariate polynomials [21]. The PCS consists of the following algorithms:

- $\mathsf{Setup}(1^\lambda, d_{\max}) \to \mathsf{pp}$: Generates public parameters supporting commitments to polynomials of degree at most $d_{\max}$.
- $\mathsf{Commit}(\mathsf{pp}, f) \to c$: Commits to a polynomial f and outputs a commitment c.
- $\mathsf{Open}(\mathsf{pp}, f, z) \to (v, \pi)$: Given a polynomial f and an evaluation point z, outputs the evaluation $v = f(z)$ together with an opening proof π.
- $\mathsf{Verify}(\mathsf{pp}, c, z, v, \pi) \to \{0, 1\}$: Verifies that v is the correct evaluation of the polynomial committed in c at point z.

Batched openings. In addition to single-point openings, we use a batched opening interface that allows the prover to open multiple evaluation claims together.

- $\mathsf{BatchOpen}(\mathsf{pp}, \{f_i\}_{i\in[m]}, \{Q_i\}_{i\in[m]}) \to (\{v_{i,z}\}_{i\in[m], z\in Q_i}, \mathit{\Pi})$: For each committed polynomial f_i and each query point set Q_i, outputs all claimed evaluations $v_{i,z} = f_i(z)$ together with a single batched proof $\mathit{\Pi}$.
- $\mathsf{BatchVerify}(\mathsf{pp}, \{c_i\}_{i\in[m]}, \{Q_i\}_{i\in[m]}, \{v_{i,z}\}_{i\in[m], z\in Q_i}, \mathit{\Pi}) \to \{0, 1\}$: Verifies that all claimed evaluations are consistent with the corresponding commitments.

Table 2. Main variables and tensor shapes used in the proof.

Variable	Shape	Meaning
$\mathbf{D}$	$[B]^{M\times d}$	Quantized training dataset.
$\mathbf{Y}$	$\{0,1\}^{K\times M}$	One-hot label matrix.
$\mathbf{S},\mathbf{G}$	$\mathbb{F}^{T\times K\times M}$	Training scores and pseudo-residuals.
$\mathbf{lc},\mathbf{rc},\mathbf{par}$	$[N_{\text{tot}}]^{N_{\text{tot}}}$	Left-child, right-child, and parent indices.
$\mathbf{N}^{\leftarrow},\mathbf{N}^{\rightarrow}$	$[B+1]^{N_{\text{tot}}\times d}$	Lower and upper node-box bounds.
$\mathbf{E}$	$\{0,1\}^{N_{\text{tot}}\times d}$	One-hot split-feature masks.
tree	$[TK+1]^{N_{\text{tot}}}$	Tree id.
depth	$\{0,\dots,h\}^{N_{\text{tot}}}$	Node depth.
score	$\mathbb{F}^{N_{\text{tot}}}$	Leaf score.
root	$\{0,1\}^{N_{\text{tot}}}$	Root indicator.
is_leaf	$\{0,1\}^{N_{\text{tot}}}$	Leaf indicator.
valid	$\{0,1\}^{N_{\text{tot}}}$	Valid-node indicator.
$\mathbf{C}_{\text{leaf}},\mathbf{V}_{\text{leaf}}$	$\mathbb{F}^{N_{\text{tot}}\times d\times B}$	Leaf count and residual-sum histograms.
$\mathbf{C},\mathbf{V}$	$\mathbb{F}^{N_{\text{tot}}\times d\times B}$	Count and residual-sum histograms aggregated from children.
$\mathbf{C}_{<},\mathbf{V}_{<},\mathbf{C}_{\geq},\mathbf{V}_{\geq}$	$\mathbb{F}^{N_{\text{tot}}\times d\times B}$	Prefix and suffix split statistics.

Properties. We rely on the following properties. *Completeness*: honestly generated openings and batched openings always verify. *Knowledge soundness*: for every polynomial-time prover producing accepted openings, there exists a polynomial-time extractor that recovers a degree-bounded polynomial for every commitment, consistent with all accepted single and batched openings. This implies evaluation binding.

2.3 Gradient Boosted Decision Trees in Plaintext

Given a commitment to a forest, the task of the proof of training is to show that the forest arises from a valid execution of the GBDT training procedure.

Let $\mathbf{D}\in[B]^{M\times d}$ be the quantized training dataset, where M is the number of samples, d is the number of features, and each feature value lies in one of B bins. Let $\mathbf{Y}\in\{0,1\}^{K\times M}$ be the one-hot label matrix, with class-sample axes, for a K-class classification task. We treat the number of boosting rounds T, the number of classes K, and the dataset size M as public parameters of the statement being proved.

Plaintext training. GBDT trains an additive model over T rounds. The cumulative scores and pseudo-residuals are arranged as tensors $\mathbf{S},\mathbf{G}\in\mathbb{F}^{T\times K\times M}$, with axes ordered by round, class, and sample. The entry $\mathbf{S}[t,k,r]$ denotes the cumulative score of sample r for class k after round t has been added, while $\mathbf{G}[t,k,r]$ denotes the regression target used by the class-k tree in round t. For convenience, we write the virtual initial score state as $\mathbf{S}[-1,k,r]=0$.

The class probabilities after round t are obtained by applying softmax over the class axis:

$$\boldsymbol{\pi}[t,k,r] := \frac{\exp(\mathbf{S}[t,k,r])}{\sum_{c\in[K]}\exp(\mathbf{S}[t,c,r])}. \tag{2.1}$$

In the fixed-point trainer, the softmax is evaluated on shifted scores $\mathbf{S}[t,k,r] - \max_{c\in[K]} \mathbf{S}[t,c,r]$. Let $\mathbf{Y} \in \{0,1\}^{K\times M}$ be the one-hot label matrix. The multinomial logistic loss after round t is

$$\mathcal{L}^{(t)} := -\sum_{r\in[M]} \sum_{k\in[K]} \mathbf{Y}[k,r] \log \boldsymbol{\pi}[t,k,r]. \tag{2.2}$$

Its partial derivative with respect to the score entry $\mathbf{S}[t,k,r]$ is

$$\frac{\partial \mathcal{L}^{(t)}}{\partial \mathbf{S}[t,k,r]} = \boldsymbol{\pi}[t,k,r] - \mathbf{Y}[k,r]. \tag{2.3}$$

Therefore, after round t, the next negative-gradient target is

$$\mathbf{G}[t+1,k,r] := -\frac{\partial \mathcal{L}^{(t)}}{\partial \mathbf{S}[t,k,r]} = \mathbf{Y}[k,r] - \boldsymbol{\pi}[t,k,r], \qquad t < T-1. \tag{2.4}$$

The initial target is set as $\mathbf{G}[0,k,r] = \mathbf{Y}[k,r] - \mathbf{1}/K$. We call $\mathbf{G}[t,\cdot,\cdot]$ the pseudo-residual for round t. As in gradient descent, we want to move the score matrix toward the negative-gradient direction and thereby reduce the loss. Thus in round t, the learner trains one regression tree for each class $k \in [K]$ to fit the target vector $\mathbf{G}[t,k,\cdot] \in \mathbb{F}^M$.[4]

Regression tree. Fix a round t and a class k. The learner constructs a regression tree $f^{(t,k)} : [B]^d \to \mathbb{F}$, which is a binary decision tree whose leaves carry scalar prediction values. Each internal node u is labeled by a feature index $j_u \in [d]$ and a threshold bin $b_u \in [B]$. Given a sample $\mathbf{x} \in [B]^d$, routing at node u is determined by the predicate $\mathbf{x}[j_u] < b_u$: if it holds, the sample is sent to the left child of u; otherwise it is sent to the right child. The output $f^{(t,k)}(\mathbf{x})$ is the value stored at the leaf reached by this routing procedure.

Similarly, during training, each node u is associated with the set $I_u \subseteq [M]$ of samples currently reaching u. The root contains all samples, i.e., $I_{\text{root}} = [M]$, and each split partitions the parent sample set into the left and right child sample sets.

How a node is split. For the class-k tree in round t, let

$$g_r := \mathbf{G}[t,k,r]$$

denote the regression target attached to sample r. Consider a node u with current sample set I_u. For a candidate split specified by a feature $j \in [d]$ and a threshold bin $b \in [B]$, define

$$L_{u,j,b} := \{\, r \in I_u : \mathbf{D}[r,j] < b \,\}, \tag{2.5}$$

$$R_{u,j,b} := \{\, r \in I_u : \mathbf{D}[r,j] \geq b \,\}. \tag{2.6}$$

[4] The above describes the training process for typical gradient-boosting decision trees. It is also possible to adapt our constructions to other variants, e.g. XGBoost [8], by modifying the training target and tree-splitting policy accordingly. It is also possible to extend our approach to prove the training and inference of random forests. See Appendices F.1 and F.2.

The quality of this split is measured by the residual sum of squares (RSS):

$$\mathrm{RSS}(u,j,b) = \sum_{r \in L_{u,j,b}} (g_r - \bar{g}_L)^2 + \sum_{r \in R_{u,j,b}} (g_r - \bar{g}_R)^2, \tag{2.7}$$

where

$$\bar{g}_L := \frac{1}{|L_{u,j,b}|} \sum_{r \in L_{u,j,b}} g_r, \qquad \bar{g}_R := \frac{1}{|R_{u,j,b}|} \sum_{r \in R_{u,j,b}} g_r$$

are the empirical means on the two sides. Intuitively, the split is good if the targets assigned to each child are well concentrated around a single mean.

If u becomes a leaf, its optimal constant prediction is

$$\bar{g}_u := |I_u|^{-1} \sum_{r \in I_u} g_r,$$

with

$$\mathrm{RSS}_{\mathrm{leaf}}(u) := \sum_{r \in I_u} (g_r - \bar{g}_u)^2.$$

Thus, the RSS reduction of a valid candidate is

$$\mathrm{Gain}(u,j,b) := \mathrm{RSS}_{\mathrm{leaf}}(u) - \mathrm{RSS}(u,j,b). \tag{2.8}$$

Among all valid candidates, i.e., those for which both children are nonempty, the trainer selects one maximizing (2.8). For a public threshold $\tau_{\mathrm{split}} \geq 0$, the node becomes a leaf if it has reached max depth h, if no valid candidate exists, or if the maximum gain is smaller than τ_{split}; otherwise, the selected split is applied. If multiple optimal eligible splits exist, we break the tie by requiring the trainer to use the one with the smallest $jB + b$ index.

Once the tree topology is fixed, each leaf ℓ is assigned the mean target value of the samples that reach it. Writing $I_\ell \subseteq [M]$ for the sample set of leaf ℓ, its output value is

$$w_\ell^{(t,k)} := \frac{1}{|I_\ell|} \sum_{r \in I_\ell} \mathbf{G}[t,k,r]. \tag{2.9}$$

For each training sample r, the prediction $f^{(t,k)}(\mathbf{D}[r,\cdot])$ is exactly the leaf value associated with the leaf reached by sample r.[5]

Round update. After training all K regression trees $\{f^{(t,k)}\}_{k \in [K]}$ in round t, the learner updates the score matrix by adding the tree outputs with learning rate $\mu > 0$:

$$\mathbf{S}[t,k,r] = \mathbf{S}[t-1,k,r] + \mu\, f^{(t,k)}(\mathbf{D}[r,\cdot]). \tag{2.10}$$

This completes round t. The next round recomputes $\boldsymbol{\pi}[t,\cdot,\cdot]$, forms the new pseudo-residual slice $\mathbf{G}[t+1,\cdot,\cdot]$ using (2.4), and trains a fresh collection of K

[5] In our later implementation, the stored leaf score is the learning-rate-scaled contribution $\mu \cdot w_\ell^{(t,k)}$ rather than the unscaled mean $w_\ell^{(t,k)}$.

regression trees against those updated targets. After T rounds, the forest consists of the TK trees trained across all rounds.

During the inference phase, for a fresh sample $\mathbf{x} \in [B]^d$, the model accumulates the per-class scores

$$\sum_{t=0}^{T-1} \mu \cdot f^{(t,k)}(\mathbf{x}) \qquad \text{for each } k \in [K],$$

and predicts from the resulting K-dimensional score vector, typically by applying softmax. On ties, any maximizing class is a valid output.

Implication for the proof. The proof-critical step is the correctness of the regression-tree training at each node, namely, that the committed split metadata and leaf values are consistent with the pseudo-residual targets of that round and class. In standard plaintext implementations, the optimizer often evaluates split candidates by sorting samples according to each feature and scanning candidate thresholds in that sorted order. In quantized systems, the same step is commonly optimized and expressed through per-bin histograms. Our proof follows the latter view: instead of proving a sorting procedure, we prove the correctness of the histogram statistics from which the trainer derives the same split scores.

2.4 Quantization

We follow the fixed-point quantization approach used in prior ZKP for ML works [25, 27, 41]. A real number x is encoded as $q_x = \lfloor x2^q \rfloor$ and interpreted as $x \approx q_x/2^q$. Multiplication is rescaled by

$$q_z = \left\lfloor \frac{q_x q_y}{2^q} \right\rfloor.$$

We write $\mu_q := \lfloor \mu 2^q \rfloor$ for the public fixed-point learning rate. For a positive even public bound $\mathcal{B}$, we assume all fixed-point operands and results fall in the signed code space $[-\mathcal{B}/2, \mathcal{B}/2) \cap \mathbb{Z}$, canonically embedded in $\mathbb{F}_p$; nonnegative remainders and comparison slacks use $[0, \mathcal{B})$. We require

$$\begin{aligned} p > 8 \max\big\{ & 2^L,\ dB,\ T\mathcal{B},\ dBM^2\mathcal{B}^2, \\ & dM\mathcal{B} \max\{2^q, |\mu_q|\},\ K2^{2q} \big\}. \end{aligned} \tag{2.11}$$

The public bound $\mathcal{B}$ is chosen to contain all honest comparison slacks, including the combined gain-and-index expression used in the split-optimality check. Thus every integer expression in the aggregation, division, and comparison checks has magnitude below $p/2$, so equality in $\mathbb{F}_p$ agrees with integer equality.[6]

Section 2.3 presents the real-valued objective for intuition. TERRAE certifies its public fixed-point instantiation, with ties in the quantized split score resolved by the smallest flattened index $jB + b$.

[6] This arithmetic condition is independent of the soundness requirement. See Appendix I and Equation (I.7) for soundness analysis and requirements.

2.5 Table Lookup Argument

Given a matrix $\mathbf{F} \in \mathbb{F}^{n \times m}$ and a matrix $\mathbf{T} \in \mathbb{F}^{N \times m}$, the lookup relation

$$\mathbf{F} \subseteq \mathbf{T}$$

means that every row of $\mathbf{F}$ appears as a row of $\mathbf{T}$.

We use a batched variant for $\mathbf{F}_i \in \mathbb{F}^{n \times m_i}$ and $\mathbf{T}_i \in \mathbb{F}^{N \times m_i}$, $i \in [k]$, proving

$$\mathbf{F}_0 \| \cdots \| \mathbf{F}_{k-1} \subseteq \mathbf{T}_0 \| \cdots \| \mathbf{T}_{k-1}.$$

Padding the matrices to equal width and randomly combining their columns reduces this to one ordinary table lookup. See Appendix G.1.

After column compression, we use two single-column lookup arguments: with preprocessing, and without preprocessing.

With preprocessing. Using the KZG PCS [21], Campanelli et al. [7] give a zero-knowledge argument for this relation where the online prover time is independent of $\mathbf{T}$. It reduces the table to one column, applies LogUp [19], and caches its quotient [11]. The only drawback is that the preprocessing requires extra work, and the advantage is notable only when $\mathbf{T}$ is large. It is knowledge-sound in the AGM and ROM under the corresponding PDL assumption [7].

Without preprocessing. For table lookup without preprocessing, we use LogUp [19] with Aurora's univariate sum-check [3] and KZG commitments. The benefit of using this construction is that, as we will demonstrate later, we can batch all lookup instances for all (differently shaped) tables into one single lookup argument, and therefore achieve a significant reduction in proof size and prover time. The efficiency and proof size gain is enormous when the table is small.[7]

Range checks. For a public bit bound ρ, TERRAE proves $0 \leq x < 2^\rho$ using $\ell = \lceil \rho / w \rceil$ base-2^w limbs. It proves

$$\begin{aligned} x &= \sum_{i=0}^{\ell-1} 2^{wi} x_i, \\ 0 &\leq x_i < 2^{\min\{w, \rho - iw\}}. \end{aligned}$$

Thus the most significant limb uses the remaining bit width. All range bounds used below are instantiated as powers of two. For clarity of demonstration, we use this range check as a primitive in the following, instead of explicitly writing the limb-decomposition and each individual lookup constraint.

[7] We do not know whether it is possible to batch the lookup arguments with preprocessing, because the preprocessing is specific to each table and cannot be easily aligned across different tables. We leave this question for future work.

3 Threat Model and Security Statement

3.1 Threat Model and Security Goal

We consider a prover that trains or serves a GBDT model for a verifier. The prover may be malicious: it may commit to an arbitrary (ill-formed) forest or claim an incorrect inference output, and it may try to convince the verifier that these objects arise from the public quantized training and inference algorithms. The verifier is probabilistic polynomial time and checks only the commitments, public parameters, public inputs, and proof transcript.

The protocol is organized into three phases.

- **Committing phase.** The prover commits to the training dataset and the trained forest.
- **Forest phase.** The prover proves that the committed forest is well formed and, when training certification is requested, that it is consistent with a valid execution of the public GBDT trainer on the committed dataset.
- **Inference phase.** Given a batch of public inputs, the prover proves that the claimed inference output is consistent with the committed forest.

In the proof of training, the public statement consists of the parameters that fix the training procedure, including the number of rounds T, the number of classes K, the dataset size M, the feature dimension d, and the number of bins B. The (optionally private) witness consists of the committed dataset, labels, forest, and auxiliary training trace. The forest includes the split metadata, tree topology, leaf boxes, and leaf values of all trees. Based on the application, the dataset and the forest may be private or public.

Theorem 3.1 (Knowledge soundness of Terrae). *For polynomially bounded parameters and an honestly generated, degree-bounded KZG SRS, work in the AGM [12] and model Fiat–Shamir as a random oracle. Assume the degree-parameterized PDL assumption used by CQ++ [7], the knowledge soundness of both lookup instantiations in Section 2.5, and extractability of the underlying KZG PCS, including the batched openings. Assume also that the field size guarantees the no-wrap-around condition in Equation* (2.11) *and provides a sufficient soundness budget.*[8] *Then:*

1. ***Completeness.*** *If the committed forest is obtained by honestly running the public quantized trainer on the committed dataset, then the prover can produce an accepting proof of training. If the committed forest is well-formed, the prover can produce an accepting forest-well-formedness proof and an inference proof for any public input batch and its correct output.*
2. ***Knowledge soundness.*** *For every probabilistic polynomial-time prover, there is a polynomial-time extractor such that, whenever the verifier accepts a proof produced by the prover, the extractor recovers degree-bounded polynomials*

[8] For clarity, we analyse the soundness in Appendix I and provide the bound in Equation (I.7).

consistent with all witness commitments and accepted openings. Except with the negligible error bounded in Appendix I, these polynomials encode a dataset, labels, forest, and training trace satisfying the public quantized trainer for a training proof; a well-formed forest for a well-formedness proof; and an inference witness consistent with the claimed output. If the same forest also has an accepting well-formedness proof, the output is a valid prediction of that forest.

Zero-knowledge. Zero knowledge is orthogonal to the theorem above. As is standard for polynomial interactive oracle proofs (PIOP), TERRAE can be compiled into a zero-knowledge protocol by masking each polynomial encoding with a random multiple of its domain's vanishing polynomial and using the zero-knowledge variants of all component arguments, including the lookup argument of Campanelli et al. [7].

4 Technical Overview

TERRAE proves GBDT training by exposing the algebra already present in histogram-based tree learning, rather than compiling the trainer into a large generic circuit. In this section, we give a high-level overview of the main technical ingredients, including how to batch claims across different domains and how to prove histogram aggregation without a RAM argument.

4.1 Batching Across Different Domains

Direct algebraic certification creates many relations over many different domains. Training uses tensors indexed by rounds, classes, samples, nodes, features, and bins; lookup, range, prefix-sum, division, maximum, and zero-check relations therefore have different lengths and shapes. Ordinary same-domain batching would force extra commitments or leave many small claims unbatched. TERRAE develops two batching tools for this setting.

Domain-lifting batching. A low-degree algebraic constraint over $\mathbb{H}_n$ is specified by a constant-arity, constant-degree polynomial $\phi(x_1, \ldots, x_k)$. Vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{F}^n$ satisfy it when

$$\forall i \in [n].\ \phi(\mathbf{v}_1[i], \ldots, \mathbf{v}_k[i]) = 0 \tag{4.1}$$

$$\iff \forall X \in \mathbb{H}_n.\ \phi(v_1(X), \ldots, v_k(X)) = 0. \tag{4.2}$$

If $v_1(X), \ldots, v_k(X)$ encode these vectors over $\mathbb{H}_n$ and are committed, the polynomial $f(X) = \phi(v_1(X), \ldots, v_k(X))$ is virtual: opening $f(t)$ reduces to opening the $v_j(t)$ values and the verifier evaluating ϕ locally. For example, $\phi(x, y) = x^2 - y$ encodes the relation $\mathbf{u} \odot \mathbf{u} = \mathbf{v}$.

Therefore every low-degree algebraic constraint becomes a zero-check for some possibly virtual $f(X)$:

$$\forall X \in \mathbb{H}_n.\ f(X) = 0. \tag{4.3}$$

We prove this zero-check through the divisibility test

$$f(X) = q(X)\nu_n(X), \quad \nu_n(X) = X^n - 1. \tag{4.4}$$

Here ν_n is the vanishing polynomial of $\mathbb{H}_n$.

Suppose two algebraic constraints live over nested power-of-two domains $\mathbb{H}_{n_1} \subset \mathbb{H}_{n_2}$. After converting them to zero-checks $f_1(X)$ over $\mathbb{H}_{n_1}$ and $f_2(X)$ over $\mathbb{H}_{n_2}$, instead of proving them separately, TERRAE lifts the smaller polynomial to the larger domain:

$$\widehat{f_1}(X) := f_1(X)\frac{\nu_{n_2}(X)}{\nu_{n_1}(X)}. \tag{4.5}$$

Note that this lifting polynomial $\nu_{n_2}(X)/\nu_{n_1}(X)$ vanishes over $\mathbb{H}_{n_2} \setminus \mathbb{H}_{n_1}$ and is non-zero for any input in $\mathbb{H}_{n_1}$. Therefore, zero-checking f_1 over $\mathbb{H}_{n_1}$ is equivalent to zero-checking $\widehat{f_1}$ over $\mathbb{H}_{n_2}$. The verifier can then batch the lifted zero-check with the larger one as

$$\widehat{f_1}(X) + \eta f_2(X) = q(X)\nu_{n_2}(X) \tag{4.6}$$

for a random challenge η.

A bonus here is that the Aurora sum-check [3] also reduces to a divisibility test, and hence we can batch sum-check claims in a similar manner. See Theorem D.1 for details.

Interleaving batching. For a non-native predicate φ, a vector $\mathbf{v} \in \mathbb{F}^n$ satisfies the predicate when

$$\forall i \in [n].\ \varphi(\mathbf{v}[i]) = 1. \tag{4.7}$$

Here φ is no longer a low-degree polynomial, so the verifier cannot derive a virtual commitment as above. Examples include table membership, range, and fixed-point division.

We batch such constraints by interleaving same-predicate instances. For two vectors $\mathbf{f}, \mathbf{g} \in \mathbb{F}^n$ satisfying the same predicate ϕ, TERRAE interleaves them into a vector

$$(\mathbf{f}[0], \mathbf{g}[0], \mathbf{f}[1], \mathbf{g}[1], \ldots) \in \mathbb{F}^{2n}. \tag{4.8}$$

In polynomial form, if f and g are encoded over $\mathbb{H}_n$, the polynomial encoding the interleaved vector is

$$\mathrm{Int}_n(f, g)(X) := \mathrm{even}_n(X)f(X) + \mathrm{odd}_n(X)g(\omega_{2n}^{-1}X), \tag{4.9}$$

where

$$\mathrm{even}_n(X) = \frac{X^n + 1}{2}, \qquad \mathrm{odd}_n(X) = \frac{1 - X^n}{2}. \tag{4.10}$$

Note that both $\mathrm{even}_n(X)$ and $\mathrm{odd}_n(X)$ are sparse and public, and the verifier can evaluate them at any point with $O(\log n)$ field operations.

For vectors $\mathbf{f} \in \mathbb{F}^n$ and $\mathbf{g} \in \mathbb{F}^{nr}$ of different lengths, we first lift the shorter vector to the larger domain by a substitution $X \mapsto X^r$, i.e. defining

$$\widehat{f}(X) := f(X^r). \tag{4.11}$$

As stated in Lemma D.3, $\widehat{f}(X)$ replicates $\mathbf{f}$ r times over the domain $\mathbb{H}_{nr}$. We then interleave $\widehat{\mathbf{f}}$ and $\mathbf{g}$ and derive the final interleaved polynomial, which is still virtual: its openings are derived from evaluations of existing commitments and public sparse selectors, so batching does not require committing to new auxiliary witness polynomials.

We present the formal interleaving method in Lemma D.5 and use it to batch all non-native predicates in Terrae.

4.2 Histogram Aggregation Without RAM

GBDT training repeatedly converts sample-wise information into per-node, per-feature, and per-bin statistics. Prior certification-style systems [26, 27] naturally model this as a memory update process: for each routed sample, update the corresponding histogram bucket and prove the resulting RAM trace. Terrae instead uses a histogram-specific argument.

For simplicity, consider a simplified version of histogram aggregation. Suppose we have vectors $\mathbf{ind} \in [m]^n$, $\mathbf{u} \in \mathbb{F}^n$, and $\mathbf{v} \in \mathbb{F}^m$. Our goal is to prove that $\mathbf{v}$ is the histogram of $\mathbf{u}$ according to the indices in $\mathbf{ind}$:

$$\mathbf{v}[j] = \sum_{i:\mathbf{ind}[i]=j} \mathbf{u}[i]. \tag{4.12}$$

After the relevant commitments are fixed, the verifier samples $\alpha \leftarrow \mathbb{F}$, which binds each i-th slot to α^i. The parties next prove

$$\sum_{i=0}^{n-1} \mathbf{u}[i] \cdot \alpha^{\mathbf{ind}[i]} = \sum_{j=0}^{m-1} \mathbf{v}[j] \cdot \alpha^j. \tag{4.13}$$

The remaining question is how to prove the exponentiated terms $\alpha^{\mathbf{ind}[i]}$. The prover commits a power table $\mathbf{p} \in \mathbb{F}^m$ with intended entries $\mathbf{p}[j] = \alpha^j$. It proves that $\mathbf{p}$ is well formed by checking

$$\mathbf{p}[0] = 1, \tag{4.14}$$

$$\mathbf{p}[j] = \alpha \cdot \mathbf{p}[j-1], \qquad j \in \{1, \ldots, m-1\}. \tag{4.15}$$

Or equivalently in polynomial form,

$$p(\omega_m^0) = 1, \tag{4.16}$$

$$(p(X) - \alpha \cdot p(\omega_m^{-1} X)) \cdot (1 - \lambda_m^{(0)}(X)) = q(X)\nu_m(X), \tag{4.17}$$

where recall $\lambda_m^{(0)}(X)$ is a Lagrange basis polynomial, and the second relation is a zero-check.

The prover then proves by lookup that each row $(\mathbf{ind}[i], \mathbf{pow}[i])$ appears in the public-indexed table $(j, \mathbf{p}[j])_{j\in[m]}$, i.e.

$$\mathbf{ind} \| \mathbf{pow} \subseteq (0, 1, \ldots, m-1)^\top \| \mathbf{p}. \tag{4.18}$$

This binds $\mathbf{pow}[i] = \alpha^{\mathbf{ind}[i]}$, after which Equation (4.13) becomes

$$\sum_{i=0}^{n-1} \mathbf{u}[i] \cdot \mathbf{pow}[i] = \sum_{j=0}^{m-1} \mathbf{v}[j] \cdot \mathbf{p}[j], \tag{4.19}$$

which can be checked by a standard sum-check.

Soundness follows from the randomness of α. To analyze the soundness error introduced by Equation (4.13), define

$$\Delta_j = \mathbf{v}[j] - \sum_{i:\mathbf{ind}[i]=j} \mathbf{u}[i].$$

If the claim that $\mathbf{v}$ is the histogram of $\mathbf{u}$ is false, then some $\Delta_j \neq 0$, and the accepted aggregate identity implies

$$\sum_{j=0}^{m-1} \Delta_j \alpha^j = 0.$$

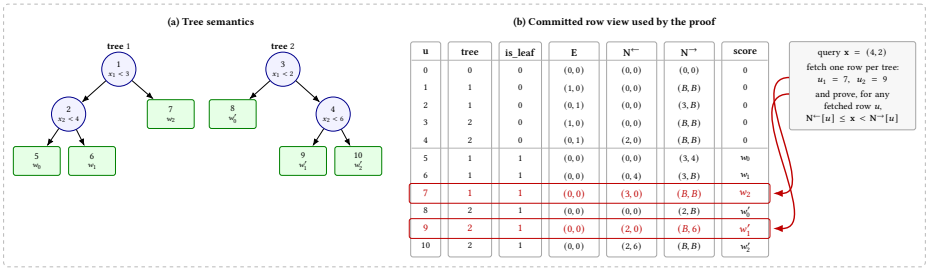
This is a nonzero polynomial in α of degree at most $m-1$, so a cheating prover can satisfy it with probability at most $(m-1)/p$, assuming α is sampled after the relevant commitments are fixed. Thus the aggregate identity is a compact randomized check of the full histogram relation.

5 Methodology

This section describes the main relations that make up the TERRAE proof system: how the forest is committed, how inference is checked, how forest well-formedness is certified, and how the training proof reduces split selection to algebraic relations over histograms.

5.1 Forest Representation

We adopt the hyperbox forest encoding of [7]. Each node carries a feature-wise half-open interval box, and a sample reaches a leaf exactly when its quantized feature vector lies inside that box. We assume there are N_{tot} nodes, and all committed objects share one global node namespace, so node identifiers are $u \in \{0, \ldots, N_{\text{tot}} - 1\}$ everywhere. Node 0 is a dummy node, and all of its committed attributes are fixed to the dummy value 0. The committed forest consists of the commitments of:



u	tree	is_leaf	E	N←	N→	score
0	0	0	(0, 0)	(0, 0)	(0, 0)	0
1	1	0	(1, 0)	(0, 0)	(B, B)	0
2	1	0	(0, 1)	(0, 0)	(3, B)	0
3	2	0	(1, 0)	(0, 0)	(B, B)	0
4	2	0	(0, 1)	(2, 0)	(B, B)	0
5	1	1	(0, 0)	(0, 0)	(3, 4)	w_0
6	1	1	(0, 0)	(0, 4)	(3, B)	w_1
7	1	1	(0, 0)	(3, 0)	(B, B)	w_2
8	2	1	(0, 0)	(0, 0)	(2, B)	w'_0
9	2	1	(0, 0)	(2, 0)	(B, 6)	w'_1
10	2	1	(0, 0)	(2, 6)	(B, B)	w'_2

Fig. 1. An illustration of the forest encoding and inference semantics. The left panel shows two compact trees with global node identifiers, and the right panel shows the committed objects used by inference. The highlighted entries are the fetched rows for one query, one per tree. Batched inference performs this fetch for every tree-sample pair in one shot and then aggregates scores across trees for each sample-class pair.

- left and right child index vectors $\mathbf{lc}, \mathbf{rc} \in [N_{\text{tot}}]^{N_{\text{tot}}}$, and parent index vector $\mathbf{par} \in [N_{\text{tot}}]^{N_{\text{tot}}}$, where invalid entries are set to 0;
- lower and upper node bounds $\mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow} \in [B+1]^{N_{\text{tot}} \times d}$, where row u stores the d-dimensional half-open box of node u and every leaf row describes one accepting region of the tree;
- a feature-selection matrix $\mathbf{E} \in \{0,1\}^{N_{\text{tot}} \times d}$, whose internal-node rows are one-hot and indicate which feature is split at each node, while leaf rows are zero;
- a tree-index vector $\mathbf{tree} \in [TK+1]^{N_{\text{tot}}}$, where tree identifier 0 is the dummy value and real trees are ranked from 1 to TK, and a root indicator $\mathbf{root} \in \{0,1\}^{N_{\text{tot}}}$;
- a depth indicator $\mathbf{depth} \in \{0,1,\ldots,h\}^{N_{\text{tot}}}$, where a root has depth 1 and h is the maximum depth of the trees in the forest. This is used for proving the correctness of tree construction.
- a leaf-score vector $\mathbf{score} \in \mathbb{F}^{N_{\text{tot}}}$, whose leaf entries store the learning-rate-scaled scalar contribution returned by that tree and whose internal-node entries are zero;
- a boolean vector $\mathbf{is_leaf} \in \{0,1\}^{N_{\text{tot}}}$, where $\mathbf{is_leaf}[u] = 1$ if and only if u is a leaf node;
- a boolean vector $\mathbf{valid} \in \{0,1\}^{N_{\text{tot}}}$, where $\mathbf{valid}[u] = 0$ iff the node is dummy.

We assume that the forest contains N_{int} valid internal nodes and N_{leaf} valid leaf nodes. The remaining committed rows are dummy padding rows; hence,

$$N_{\text{int}} + N_{\text{leaf}} \leq N_{\text{tot}} - 1,$$

where node 0 is reserved as the canonical dummy node.

Figure 1 shows the intuition behind the forest encoding: the left panel illustrates the structure of the forest, while the right panel presents the committed objects.

5.2 Proof of Inference

To prove the correctness of inference over batched input $\mathbf{I} \in \mathbb{F}^{b\times d}$, the prover commits $\mathbf{U}^{\leftarrow}, \mathbf{U}^{\rightarrow} \in \mathbb{F}^{TKb\times d}$ and $\mathbf{sc} \in \mathbb{F}^{TKb}$. Each row of $\mathbf{U}^{\leftarrow}$ and $\mathbf{U}^{\rightarrow}$ denotes the claimed lower and upper box bounds of the leaf reached by an input in a particular tree, and the corresponding entry of $\mathbf{sc}$ denotes that leaf's score. The following lookup binds these claims to the same committed leaf:

$$\mathbf{1}\|\mathbf{ind_tree}\|\mathbf{U}^{\leftarrow}\|\mathbf{U}^{\rightarrow}\|\mathbf{sc} \subseteq \mathbf{is_leaf}\|\mathbf{tree}\|\mathbf{N}^{\leftarrow}\|\mathbf{N}^{\rightarrow}\|\mathbf{score}, \tag{5.1}$$

where notation "$\subseteq$" means that each row of the left matrix equals some row of the right matrix, and

$$\mathbf{ind_tree} \coloneqq (\underbrace{1,\ldots,1}_{b},\underbrace{2,\ldots,2}_{b},\ldots,\underbrace{TK,\ldots,TK}_{b})^{\top}.$$

The pair (t,k) indicates a tree with identifier $tK+k+1$, while 0 is reserved for dummy rows. As a sanity check, the LHS of Equation (5.1) has shape $TKb \times (2d+3)$, and the RHS has shape $N_{\text{tot}} \times (2d+3)$, so the table lookup is well-defined. Padding and randomly combining the columns reduces (5.1) to the lookup of [7], with online prover time $O(TKb)$.

The lookup aligns each row of $\mathbf{U}^{\leftarrow}$, $\mathbf{U}^{\rightarrow}$, and $\mathbf{sc}$ to one leaf. Proving

$$\mathbf{U}^{\leftarrow}_{\{T\times K\times b\times d\}}[t,k,i,j] \le \mathbf{I}[i,j] < \mathbf{U}^{\rightarrow}_{\{T\times K\times b\times d\}}[t,k,i,j] \tag{5.2}$$

certifies the reached leaf. The prover then sums $\mathbf{sc}_{\{T\times K\times b\}}$ over rounds and proves the final maximum.

Complexity. The dominant cost for proving inference is one lookup row for every tree-sample pair and an interval check for every feature of that row. Hence the online prover time is

$$\tilde{O}(TKbd),$$

with lower-order terms for score aggregation and the final maximum over the K classes. The lookup, interval, sum, and maximum checks are batched and proved together, so the verifier opens only a constant number of committed polynomials at constantly many points. The KZG proof therefore contains $O(1)$ group elements. The verifier's remaining $\tilde{O}(\log(TKbd))$ arithmetic comes from evaluating public sparse and virtual polynomials and verifying the batched subprotocols. This online bound assumes that the input batch is supplied through its committed polynomial encoding; reading and processing bd raw public values requires $\Omega(bd)$ input processing and is excluded.

5.3 Proof of Forest Well-formedness

First, the prover proves that the committed forest is well-formed, by showing

1. Consistency between indicators, i.e. for any node i,

$$\mathbf{valid}[i] = 1 \iff \mathbf{depth}[i] \neq 0, \tag{5.3}$$
$$\mathbf{valid}[i] = 1 \iff \mathbf{tree}[i] \neq 0, \tag{5.4}$$
$$\mathbf{valid}[i] = 0 \implies \mathbf{is_leaf}[i] = 0. \tag{5.5}$$

2. The tree-index and max-depth ranges are respected, i.e.

$$\begin{aligned} \mathbf{tree}[u] &\in \{0, \ldots, TK\}, \\ \mathbf{depth}[u] &\in \{0, \ldots, h\}, \qquad \forall u \in [N_{\text{tot}}]. \end{aligned} \tag{5.6}$$

3. Indicators are boolean, i.e.

$$\mathbf{root} \odot (\mathbf{root} - \mathbf{1}) = \mathbf{0}, \tag{5.7}$$
$$\mathbf{is_leaf} \odot (\mathbf{is_leaf} - \mathbf{1}) = \mathbf{0}, \tag{5.8}$$
$$\mathbf{valid} \odot (\mathbf{valid} - \mathbf{1}) = \mathbf{0}. \tag{5.9}$$

4. No ghost nodes, i.e. all invalid nodes u with $\mathbf{valid}[u] = 0$ are dummy, with

$$\mathbf{lc}[u] = \mathbf{rc}[u] = 0, \tag{5.10}$$
$$\forall j \in [d].\ \mathbf{N}^{\leftarrow}[u, j] = \mathbf{N}^{\rightarrow}[u, j] = 0, \tag{5.11}$$
$$\mathbf{tree}[u] = \mathbf{root}[u] = \mathbf{is_leaf}[u] = 0. \tag{5.12}$$

5. There are exactly TK trees, i.e.

$$\sum_{u=0}^{N_{\text{tot}}-1} \mathbf{root}[u] = TK. \tag{5.13}$$

6. The depth of each root is 1, i.e.

$$\mathbf{root} \odot (\mathbf{depth} - \mathbf{1}) = \mathbf{0}. \tag{5.14}$$

7. Roots are unique, i.e. for any distinct nodes $u \neq v$,

$$\mathbf{root}[u] = \mathbf{root}[v] = 1 \implies \mathbf{tree}[u] \neq \mathbf{tree}[v]. \tag{5.15}$$

8. The consistency between parent and child, i.e.

$$\mathbf{par}[\mathbf{lc}[u]] = \mathbf{par}[\mathbf{rc}[u]] = u, \tag{5.16}$$
$$(\mathbf{lc}[\mathbf{par}[v]] - v)(\mathbf{rc}[\mathbf{par}[v]] - v) = 0, \tag{5.17}$$

   for any valid internal node u and valid non-root node v. Also, the parent of each root is dummy, i.e.

$$\mathbf{par} \odot \mathbf{root} = \mathbf{0}. \tag{5.18}$$

9. The left and right children do not overlap, share the same tree index with the parent, and have correct depth,

$$\mathbf{lc}[u] \neq \mathbf{rc}[u], \tag{5.19}$$
$$\mathbf{tree}[\mathbf{lc}[u]] = \mathbf{tree}[\mathbf{rc}[u]] = \mathbf{tree}[u], \tag{5.20}$$
$$\mathbf{depth}[\mathbf{lc}[u]] = \mathbf{depth}[\mathbf{rc}[u]] = \mathbf{depth}[u] + 1, \tag{5.21}$$

for all internal nodes u.

10. Leaves have no children, i.e.

$$\mathbf{lc} \odot \mathbf{is_leaf} = \mathbf{0} \wedge \mathbf{rc} \odot \mathbf{is_leaf} = \mathbf{0}, \tag{5.22}$$

where "$\odot$" is the Hadamard product.

11. The hyper-box of any root is full, i.e.

$$\mathbf{Root} \odot \mathbf{N}^{\leftarrow} = \mathbf{0} \wedge \mathbf{Root} \odot \mathbf{N}^{\rightarrow} = B \cdot \mathbf{Root}, \tag{5.23}$$

where $\mathbf{Root} = (\mathbf{root} \| \cdots \| \mathbf{root})$.

12. Valid nodes have non-empty hyper-box, i.e. for every node u with non-zero depth,

$$\mathbf{N}^{\rightarrow}[u, j] - \mathbf{N}^{\leftarrow}[u, j] > 0, \quad \forall j \in [d]. \tag{5.24}$$

13. The hyper-boxes of the left and right children are consistent with the parent and the split, i.e. for each internal node u and feature j,

$$\mathbf{N}^{\leftarrow}[u, j] - \mathbf{N}^{\leftarrow}[\mathbf{lc}[u], j] = 0 \wedge \tag{5.25}$$
$$\mathbf{N}^{\rightarrow}[u, j] - \mathbf{N}^{\rightarrow}[\mathbf{rc}[u], j] = 0 \wedge \tag{5.26}$$
$$\mathbf{E}[u, j](\mathbf{N}^{\rightarrow}[\mathbf{lc}[u], j] - \mathbf{N}^{\leftarrow}[\mathbf{rc}[u], j]) = 0 \wedge \tag{5.27}$$
$$(1 - \mathbf{E}[u, j])(\mathbf{N}^{\leftarrow}[u, j] - \mathbf{N}^{\leftarrow}[\mathbf{rc}[u], j]) = 0 \wedge \tag{5.28}$$
$$(1 - \mathbf{E}[u, j])(\mathbf{N}^{\rightarrow}[u, j] - \mathbf{N}^{\rightarrow}[\mathbf{lc}[u], j]) = 0. \tag{5.29}$$

These equations make the children inherit the parent's box, partitioned at the selected feature and threshold.

14. $\mathbf{E}$ is boolean and one-hot on internal nodes and zero on leaves, i.e. for each node u,

$$\sum_{j \in [d]} \mathbf{E}[u, j] - \mathbf{valid}[u](1 - \mathbf{is_leaf}[u]) = 0, \tag{5.30}$$
$$\mathbf{E} \odot (\mathbf{E} - \mathbf{1}) = \mathbf{0}. \tag{5.31}$$

15. **score** is zero on internal nodes, i.e.

$$\mathbf{score} \odot (\mathbf{1} - \mathbf{is_leaf}) = \mathbf{0}. \tag{5.32}$$

The above involves three kinds of proving techniques: routing, sum, and histogram arguments. As an example, to prove the consistency of tree indices between nodes and their left children, the prover first commits **tree_lc** where

$$\mathbf{tree_lc}[u] = \mathbf{tree}[\mathbf{lc}[u]]$$

for all internal nodes u. It then proves

$$\mathbf{lc} \| \mathbf{tree_lc} \subseteq \begin{pmatrix} 0 \\ 1 \\ \vdots \\ N_{\text{tot}} - 1 \end{pmatrix} \| \mathbf{tree},$$

alongside

$$\mathbf{valid} \odot (\mathbf{1} - \mathbf{is_leaf}) \odot (\mathbf{tree_lc} - \mathbf{tree}) = \mathbf{0}.$$

For sums over vectors, we use Aurora's sum-check protocol [3]. To prove a sum along one dimension of the matrix, we adopt a Quarks-style proof [31], where the prover commits the prefix sum and total sum, and proves the consistency between them and the original matrix.

Additionally, to prove the uniqueness of roots, we use an approach similar to our proof of histogram. The prover commits a vector $\mathbf{cnt} \in \{0,1\}^{TK+1}$, where $\mathbf{cnt}[t]$ counts the number of roots of tree t. After proving its booleanity, the prover proves that for a verifier challenge α,

$$\sum_{t=1}^{TK} \alpha^t = \sum_{t \in [TK+1]} \alpha^t \cdot \mathbf{cnt}[t] = \sum_{u \in [N_{\text{tot}}]} \alpha^{\mathbf{tree}[u]} \cdot \mathbf{root}[u]. \tag{5.33}$$

The LHS is public and can be computed efficiently. To prove the α-exponent, the prover commits a vector of powers of α and then proves the lookup relation.

Complexity. All forest-structural checks are low-degree algebraic constraints or non-native predicates over either N_{tot} node rows or $N_{\text{tot}}d$ node-feature rows. The only additional global object is the root-count vector of length $TK + 1$, where $TK + 1 \leq N_{\text{tot}}$. Thus, the prover time for forest well-formedness is

$$\tilde{O}(N_{\text{tot}}d).$$

Domain-lifting batching combines the zero-check and sum-check claims, and interleaving batching combines the local lookup and range-check predicates. As there are only constantly many polynomial commitments, the verifier time is $\tilde{O}(\log(N_{\text{tot}}d))$.

5.4 Proof of Training

The training proof reuses the same committed forest. The prover first commits the dataset $\mathbf{D} \in [B]^{M \times d}$ and its one-hot label $\mathbf{Y} \in \{0,1\}^{K \times M}$. The prover then commits the leaves reached by each sample in the dataset, by committing

$$\mathbf{Ind} \in [N_{\text{tot}}]^{T \times K \times M},$$

where $\mathbf{Ind}[t, k, r]$ is the index of the leaf reached by sample r in the tree trained for class k at round t.

Likewise, the prover also commits the pseudo-residuals $\mathbf{G} \in \mathbb{F}^{T\times K\times M}$ and the current scores $\mathbf{S} \in \mathbb{F}^{T\times K\times M}$. For round t, class k, and sample r, $\mathbf{G}[t, k, r]$ represents the pseudo-residual target of sample r that should be approximated by the regression tree for class k at round t, and $\mathbf{S}[t, k, r]$ represents the cumulative score of sample r for class k after round t has been added. Initially, $\mathbf{G}[0, k, r] = 2^q\mathbf{Y}[k, r] - \lfloor 2^q/K \rfloor$ for all k and r. The committed leaf score stores the learning-rate-scaled tree contribution, so the public learning rate μ is folded into the leaf-value relation.

Logically, the proof of training sequentially proves that:

1. **Ind** correctly commits the leaf reached by each sample in the dataset. The proof is similar to the inference proof over $\mathbf{D}$.
2. $\mathbf{G}[0, \cdot, \cdot]$ is set correctly from $\mathbf{Y}$: $\mathbf{G}[0, k, r] = 2^q\mathbf{Y}[k, r] - \lfloor 2^q/K \rfloor$.
3. The tree trained at each round t is the optimal regression tree that fits $\mathbf{G}[t, \cdot, \cdot]$ on the dataset, i.e. each internal-node split is optimal, every leaf satisfies the stopping rule, and leaf scores are assigned correctly. This is discussed in more detail in the next subsections.
4. For each round t, the score $\mathbf{S}[t, \cdot, \cdot]$ is updated correctly from the previous score state and the scores each sample obtained from the trees trained at round t.
5. For each round $t < T - 1$, $\mathbf{G}[t + 1, \cdot, \cdot]$ is updated correctly from $\mathbf{Y}$ and $\mathbf{S}[t, \cdot, \cdot]$. This involves proving softmax over the second dimension of $\mathbf{S}$.

Together, these conditions imply that the committed objects encode a valid plaintext GBDT training trace across all rounds. Proving that each regression tree fits $\mathbf{G}[t, \cdot, \cdot]$ is the critical step in the proof, which is discussed in more detail in the next subsection.

Complexity. The training proof is dominated by three families of relations. First, routing all training samples to their committed leaves costs $\tilde{O}(TKMd)$. Second, the histogram proof processes the sample-update stream of size $TKMd$ and the committed histogram tensors of size $N_{\text{tot}}dB$. Third, split optimality and leaf-value checks scan the per-node, per-feature, and per-bin statistics, contributing $\tilde{O}(N_{\text{tot}}dB)$, while score and residual updates contribute $\tilde{O}(TKM)$ plus the softmax-related fixed-point checks over the class dimension. Overall, the prover time is

$$\tilde{O}\big(TKMd + N_{\text{tot}}dB\big).$$

The prover time is linear in the algebraic training trace, but the verifier does not reprocess the trace. After batching, the KZG proof has $O(1)$ group elements; verifier arithmetic is $\tilde{O}(\log(TKMd + N_{\text{tot}}dB))$ for the public/virtual polynomial evaluations and batched-subprotocol checks.

5.5 Proving the Correctness of Node Splits

For each committed node, TERRAE must prove that every valid candidate split is evaluated correctly, that an optimal split is selected, and that the selection is consistent with the committed forest. We establish these claims in five steps.

(i) Split gains from histograms. Fix a node u, feature j, and threshold bin b. Let I_u denote the set of samples reaching u. Splitting on (j, b) partitions I_u into $L_{u,j,b}, R_{u,j,b} \subseteq I_u$. For pseudo-residuals g_r, $r \in I_u$, the residual-sum-of-squares objective is

$$\mathrm{RSS}(u, j, b) = \sum_{r \in L_{u,j,b}} (g_r - \bar{g}_L)^2 + \sum_{r \in R_{u,j,b}} (g_r - \bar{g}_R)^2.$$

Expanding the squares yields

$$\mathrm{RSS}(u, j, b) = \sum_{r \in I_u} g_r^2 - \left(\frac{\left(\sum_{r \in L_{u,j,b}} g_r\right)^2}{|L_{u,j,b}|} + \frac{\left(\sum_{r \in R_{u,j,b}} g_r\right)^2}{|R_{u,j,b}|} \right). \tag{5.34}$$

The first term is independent of (j, b). Hence minimizing the RSS is equivalent to maximizing

$$\Phi(u, j, b) \coloneqq \frac{\left(\sum_{r \in L_{u,j,b}} g_r\right)^2}{|L_{u,j,b}|} + \frac{\left(\sum_{r \in R_{u,j,b}} g_r\right)^2}{|R_{u,j,b}|}. \tag{5.35}$$

If node u becomes a leaf, the corresponding score is

$$\Phi_{\mathrm{leaf}}(u) \coloneqq \frac{\left(\sum_{r \in I_u} g_r\right)^2}{|I_u|}.$$

As the terms $\sum_{r \in I_u} g_r^2$ cancel, the gain in (2.8) becomes

$$\mathrm{Gain}(u, j, b) = \Phi(u, j, b) - \Phi_{\mathrm{leaf}}(u).$$

Let $\mathbf{C}[u, j, k]$ count the samples at u whose j-th feature lies in bin k, and let $\mathbf{V}[u, j, k]$ be their fixed-point residual sum. These histograms encode (5.35) as

$$\Gamma(u, j, b) \coloneqq \left\lfloor \frac{\left(\sum_{k=0}^{b-1} \mathbf{V}[u, j, k]\right)^2}{2^q \cdot \sum_{k=0}^{b-1} \mathbf{C}[u, j, k]} \right\rfloor + \left\lfloor \frac{\left(\sum_{k=b}^{B-1} \mathbf{V}[u, j, k]\right)^2}{2^q \cdot \sum_{k=b}^{B-1} \mathbf{C}[u, j, k]} \right\rfloor. \tag{5.36}$$

For an unsplit node, define

$$\Gamma_{\mathrm{leaf}}(u) \coloneqq \left\lfloor \frac{\left(\sum_{k \in [B]} \mathbf{V}[u, j, k]\right)^2}{2^q \sum_{k \in [B]} \mathbf{C}[u, j, k]} \right\rfloor.$$

This value is independent of j. The factor 2^q accounts for the quantized numerators and unscaled counts. Writing $\tau_q \coloneqq \lfloor 2^q \tau_{\text{split}} \rfloor$, the trainer splits exactly when the best valid candidate satisfies $\mathit{\Gamma}(u,j,b) - \mathit{\Gamma}_{\text{leaf}}(u) \geq \tau_q$. Therefore, histograms are sufficient proof objects: prefix and suffix sums of $\mathbf{C}$ and $\mathbf{V}$ determine every valid candidate score.

(ii) Valid candidates and their scores. For the clarity of demonstration, assume that $\mathbf{C}$ and $\mathbf{V}$ have already been committed and proved correct.[9] The prover also commits the prefix and suffix sums and the node totals, and proves

$$\mathbf{C}_{<}[u,j,b] = \sum_{k=0}^{b-1} \mathbf{C}[u,j,k], \tag{5.37}$$

$$\mathbf{C}_{\geq}[u,j,b] = \sum_{k=b}^{B-1} \mathbf{C}[u,j,k], \tag{5.38}$$

$$\mathbf{C}'[u] = d^{-1} \sum_{j\in[d]} \sum_{k\in[B]} \mathbf{C}[u,j,k], \tag{5.39}$$

with analogous equations for $\mathbf{V}_{<}$, $\mathbf{V}_{\geq}$, and $\mathbf{V}'$.

The prover then commits a validity indicator $\boldsymbol{\chi} \in \{0,1\}^{N_{\text{tot}}\times d\times B}$ that records whether (u,j,b) yields a valid split. We call a split (structurally) valid if it yields two non-empty children. Using a lookup argument, it proves

$$\boldsymbol{\chi}[u,j,b] = 1 \iff \mathbf{C}_{<}[u,j,b] > 0 \wedge \mathbf{C}_{\geq}[u,j,b] > 0. \tag{5.40}$$

To prove the candidate scores, the prover commits $\boldsymbol{\mathit{\Gamma}}_{<}$, $\boldsymbol{\mathit{\Gamma}}_{\geq}$, and $\boldsymbol{\mathit{\Gamma}}$. Using lookups, it proves the entry-wise relations

$$\boldsymbol{\mathit{\Gamma}}_{<} = \left\lfloor \frac{(\mathbf{V}_{<})^2}{2^q \mathbf{C}_{<}} \right\rfloor,\ \boldsymbol{\mathit{\Gamma}}_{\geq} = \left\lfloor \frac{(\mathbf{V}_{\geq})^2}{2^q \mathbf{C}_{\geq}} \right\rfloor,\ \boldsymbol{\mathit{\Gamma}} = \boldsymbol{\mathit{\Gamma}}_{<} + \boldsymbol{\mathit{\Gamma}}_{\geq}. \tag{5.41}$$

The divisions are checked only at entries with $\boldsymbol{\chi}[u,j,b] = 1$, avoiding division by zero.

The prover similarly commits $\boldsymbol{\mathit{\Gamma}}_{\text{leaf}}$ and proves, on valid nodes,

$$\boldsymbol{\mathit{\Gamma}}_{\text{leaf}} = \left\lfloor \frac{(\mathbf{V}')^2}{2^q \mathbf{C}'} \right\rfloor. \tag{5.42}$$

Thus $\boldsymbol{\mathit{\Gamma}}[u,j,b] - \boldsymbol{\mathit{\Gamma}}_{\text{leaf}}[u]$ is the quantized gain of splitting node u by (j,b) whenever $\boldsymbol{\chi}[u,j,b] = 1$.

[9] At a high level, the prover uses **Ind** to prove the leaf histograms $\mathbf{C}_{\text{leaf}}$ and $\mathbf{V}_{\text{leaf}}$, and aggregates them from leaves to roots to obtain $\mathbf{C}$ and $\mathbf{V}$. See Appendix E.4 for details.

(iii) Selecting optimal splits. Write $\mathbf{internal} \coloneqq \mathbf{valid} \odot (\mathbf{1} - \mathbf{is_leaf})$ for the internal-node indicator. To select splits, the prover commits $\mathbf{T} \in \{0,1\}^{N_{\text{tot}} \times d \times B}$ and proves that it is Boolean and satisfies

$$\sum_{j \in [d]} \sum_{b \in [B]} \mathbf{T}[u, j, b] = \mathbf{internal}[u]. \tag{5.43}$$

Thus it is one-hot on internal nodes and zero elsewhere, which for each internal node selects exactly one feature-threshold pair. The prover then commits the purported optimal splits $\boldsymbol{\Gamma}^* \in \mathbb{F}^{N_{\text{tot}}}$, and proves

$$\boldsymbol{\Gamma}^*[u] = \sum_{j \in [d]} \sum_{b \in [B]} \mathbf{T}[u, j, b] \boldsymbol{\Gamma}[u, j, b]. \tag{5.44}$$

(iv) Optimality and stopping. For every internal node u and valid candidate (j, b), the prover proves that the selected score is maximal:

$$\begin{aligned} 0 \le dB\big(\boldsymbol{\Gamma}^*[u] - \boldsymbol{\Gamma}[u, j, b]\big) + jB + b \\ - \sum_{j' \in [d]} \sum_{b' \in [B]} \mathbf{T}[u, j', b'](j'B + b') < \mathcal{B}. \end{aligned} \tag{5.45}$$

Since candidate indices differ by less than dB, this constraint maximizes the quantized score and selects the smallest flattened index among ties. The prover then commits a Boolean split eligibility mask $\boldsymbol{\zeta}$ of the same shape as $\boldsymbol{\chi}$ and proves

$$\begin{aligned} \boldsymbol{\zeta}[u, j, b] = 1 \quad \Longleftrightarrow \quad & \boldsymbol{\chi}[u, j, b] = 1 \\ & \wedge \ \boldsymbol{\Gamma}[u, j, b] \ge \boldsymbol{\Gamma}_{\text{leaf}}[u] + \tau_q. \end{aligned} \tag{5.46}$$

Thus $\boldsymbol{\zeta}$ indicates eligible splits, which are valid and yield a gain meeting the public threshold. The prover proves that every internal node has selected one eligible split:

$$\sum_{j \in [d]} \sum_{b \in [B]} \mathbf{T}[u, j, b] \boldsymbol{\zeta}[u, j, b] = \mathbf{internal}[u]. \tag{5.47}$$

To prove the stopping condition, the prover commits a depth indicator $\mathbf{d}_{<h} \in \{0,1\}^{N_{\text{tot}}}$, and proves its correctness:

$$\mathbf{depth} \| \mathbf{d}_{<h} \subseteq \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ \vdots & \vdots \\ h-1 & 1 \\ h & 0 \end{pmatrix}. \tag{5.48}$$

Thus $\mathbf{d}_{<h}[u] = 1$ exactly when $\mathbf{depth}[u] < h$. The stopping condition is, for every u,

$$\mathbf{is_leaf}[u] \mathbf{d}_{<h}[u] \sum_{j \in [d]} \sum_{b \in [B]} \boldsymbol{\zeta}[u, j, b] = 0. \tag{5.49}$$

Equation (5.49) means that a leaf has either reached the maximum depth or has no eligible split. Conversely, (5.47) means that every internal node selects an eligible split, which is optimal by (5.45); forest well-formedness already forces its depth below h.

With (5.46) ensuring every eligible split is honestly marked, these together enforce the optimality and stopping condition: for every internal node, an optimal split is selected, and a leaf either has reached the maximum depth or has no eligible split.

(v) Binding to the committed forest. Recall that $\mathbf{E} \in \{0,1\}^{N_{\text{tot}} \times d}$ is part of the forest commitment; well-formedness proves that it is one-hot on internal nodes, where $\mathbf{E}[u,j] = 1$ selects feature j. To certify its consistency against $\mathbf{T}$, the prover proves

$$\mathbf{E}[u,j] = \sum_{b \in [B]} \mathbf{T}[u,j,b]. \tag{5.50}$$

Finally, the prover binds $\mathbf{T}$ directly to the hyper-box encoding of the committed forest. It proves

$$\mathbf{internal}[u]\mathbf{E}[u,j]\big(\mathbf{N}^{\rightarrow}[\mathbf{lc}[u],j] - \sum_{b \in [B]} b\,\mathbf{T}[u,j,b]\big) = 0, \tag{5.51}$$

$$\mathbf{internal}[u]\mathbf{E}[u,j]\big(\mathbf{N}^{\leftarrow}[\mathbf{rc}[u],j] - \sum_{b \in [B]} b\,\mathbf{T}[u,j,b]\big) = 0. \tag{5.52}$$

Together, these checks prove that all candidate gains are computed correctly, the selected splits are optimal, nodes become leaves only when the stopping condition holds, and the selected splits are consistent with the committed forest, which completes the proof. The complete commitment, lookup, and zero-check constraints are given in Appendix E.4.

6 Evaluation

6.1 Evaluation Setup

Hardware. All experiments run on Ubuntu 24.04.4 LTS with Linux kernel 6.17.0-35-generic. The machine has two Intel Xeon Gold 6326 CPUs at 2.90 GHz, with 32 physical cores, 64 logical threads, and 188 GiB of RAM.

Baseline. We use CertXGB [25], the backend-independent certification strategy of ZKBoost, as our baseline. We preserve its public full-tree padding, per-sample tree evaluation, leaf-to-root histogram reconstruction, leaf and split validation, and separation of tree-local checks from cross-tree score consistency. Following the ZKBoost v3 update, we also adopt its weighted-LogUp argument for leaf-histogram construction. For K-class classification, we use K class-specific trees

per iteration and retain the same tree-local validation structure; inference proves routing, score aggregation, and the final argmax.

We instantiate CertXGB with Plonk [14, 15] over the same KZG backend as TERRAE, adapting its path and comparison checks to tagged, grouped-limb lookups and batching all opening claims into one KZG proof. We do not carry over verifiable pre-binning (inputs are already quantized), tree-parallel execution (our experiments are single-threaded), or ZKBoost's VOLE-specific commitment, correlation-generation, and communication optimizations. Thus, the results compare common-KZG implementations rather than TERRAE with native VOLE-based ZKBoost.

Datasets and Training. Dry Bean [22] has 13,611 samples, 16 features, and 7 classes (10,891/2,720 train/test). From Optdigits [2], we retain labels 0–6, giving Optdigit* with 2,674/1,264 train/test samples and 64 features. Thus, the cross-dataset difference largely reflects Optdigit*'s fourfold larger feature dimension. We quantize features into $B = 8$ bins. The basic setting is $M = b = 512$, $h = 2$, and $T = 1$. Each sweep varies T, h, or M, with $b = M$ throughout. The Optdigit* $M = 4096$ setting uses all 2,674 real training rows and deterministically pads the proof trace to 4,096 rows by cycling through them. We set $\mu = 4$ and $\tau_{\text{split}} = 1/4$ throughout.

Implementation. Both systems are implemented in C++. Cryptographic operations use the mcl library [34]; the proof systems are instantiated over the 255-bit BLS12-381 scalar field with KZG polynomial commitments. The field has 2-adicity 32, so we use $L \leq 32$. The artifact instantiates the range-check limb width with $w = 16$. The proofs are made non-interactive with Fiat–Shamir. Both systems run single-threaded. The implementation and benchmark scripts are available at `https://github.com/GJCPP/Terrae`.

Proof scope. Training includes the proof of well-formedness and the proof that the forest is derived from running the public trainer over the committed dataset. Inference includes routing, score aggregation, and argmax for both systems and assumes a previously accepted well-formedness proof for the same forest. All auxiliary commitments are derived mechanically from the public training configuration and committed trace, so the protocol introduces no additional deployment-time parameter tuning.

Scope of comparison. For each configuration, both systems certify the same quantized dataset, forest, training and inference traces, and claimed outputs. The runtime results compare TERRAE with CertXGB under the same KZG backend; they do not compare KZG-based TERRAE with the original VOLE-based ZKBoost implementation. As the two systems adopt different proving approaches, performance differences may still arise from circuit compilation and engineering, so we interpret the measured acceleration primarily, but not exclusively, as a proof-strategy improvement.

Table 3. Asymptotic prover costs for tree-model training and inference certification. Here h is the maximum root-to-leaf depth, and N_{tot} is the number of nodes in the corresponding encoded forest; CertXGB uses full-tree padding. The $\tilde{O}(\cdot)$ notation hides logarithmic factors and fixed backend-dependent constants. Both online verifiers run in polylogarithmic time over committed polynomial inputs.

System	Proof of training	Proof of inference	Proof size
CertXGB	$\tilde{O}\big(TKM(d+h) + N_{\mathrm{tot}}dB\big)$	$\tilde{O}\big(TKb(d+h)\big)$	$O(1)$
TERRAE	$\tilde{O}\big(TKMd + N_{\mathrm{tot}}dB\big)$	$\tilde{O}(TKbd)$	$O(1)$

6.2 Asymptotic Comparison with Prior Works

Table 3 summarizes the asymptotic prover costs of TERRAE and CertXGB.

CertXGB validates, for every tree–sample pair, a feature permutation of length d and h branch decisions, whereas TERRAE's routing relation has no explicit path-length term. This changes the per-sample routing cost from $\tilde{O}(d+h)$ to $\tilde{O}(d)$. The measured end-to-end advantage additionally reflects TERRAE's compact forest encoding and batching, while CertXGB pads each tree to full depth.

6.3 GBDT Benchmark

Tables 5 and 6 use one-parameter sweeps around a minimal setting. Their purpose is to demonstrate the scaling behavior of TERRAE and CertXGB.

Table 4. Proof sizes and maximum verifier times across completed runs.

System	Stage	Proof size (KiB)	Verifier time (s)
TERRAE	Training	$\leq 30.95^{\dagger}$	≈ 0.453
CertXGB	Training	7.22	≈ 0.065
TERRAE	Inference	$\leq 4.25^{\dagger}$	≈ 0.075
CertXGB	Inference	1.15	≈ 0.008

†Maximum observed value; setting-dependent range-check batching changes training and inference proof sizes by at most 1.14 and 0.06 KiB, respectively.

Proof size and verification. Table 4 reports proof sizes and maximum verifier times. TERRAE's proofs are larger and verification is slower than CertXGB's for both training and inference.

Scaling with the number of iterations. Training time grows approximately linearly with T. On Dry Bean, TERRAE's training time rises from 32.08 s at $T = 1$ to

Table 5. Scaling results on Dry Bean. For both systems, training includes forest well-formedness.

Setting	Training					Inference				
	Time (s)			Memory (GiB)		Time (s)			Memory (GiB)	
	TERRAE	CertXGB	Speedup	TERRAE	CertXGB	TERRAE	CertXGB	Speedup	TERRAE	CertXGB
$T = 1$	32.08	211.07	6.58×	2.19	10.40	9.81	36.73	3.75×	0.81	2.19
$T = 2$	62.36	463.99	7.44×	4.52	20.60	18.83	36.87	1.96×	1.52	2.18
$T = 4$	126.56	1030.83	8.15×	9.63	41.07	36.77	81.52	2.22×	2.85	4.27
$h = 2$	32.08	211.07	6.58×	2.19	10.40	9.81	36.73	3.75×	0.81	2.19
$h = 3$	37.43	326.99	8.74×	2.62	19.51	9.78	36.18	3.70×	0.82	2.16
$h = 4$	54.48	638.58	11.72×	4.05	37.96	9.88	36.43	3.69×	0.83	2.19
$h = 5$	91.00	1115.48	12.26×	7.31	73.70	10.42	35.77	3.43×	0.85	2.18
$M = 512$	32.08	211.07	6.58×	2.19	10.40	9.81	36.73	3.75×	0.81	2.19
$M = 1024$	71.31	367.30	5.15×	5.89	19.72	19.74	82.78	4.19×	1.72	4.24
$M = 2048$	167.31	714.96	4.27×	15.67	38.61	40.79	175.85	4.31×	3.84	8.42
$M = 4096$	447.10	1423.02	3.18×	47.09	73.58	89.83	385.40	4.29×	9.67	16.79

Table 6. Scaling results on Optdigit*. For both systems, training includes forest well-formedness. A dash indicates an unfinished run due to memory limitation.

Setting	Training					Inference				
	Time (s)			Memory (GiB)		Time (s)			Memory (GiB)	
	TERRAE	CertXGB	Speedup	TERRAE	CertXGB	TERRAE	CertXGB	Speedup	TERRAE	CertXGB
$T = 1$	98.35	884.60	8.99×	6.50	39.94	35.96	79.41	2.21×	2.83	4.16
$T = 2$	202.64	1924.58	9.50×	13.56	79.47	73.86	176.43	2.39×	5.54	8.27
$T = 4$	432.93	3907.50	9.03×	29.51	157.74	156.98	396.70	2.53×	10.90	16.63
$h = 2$	98.35	884.60	8.99×	6.50	39.94	35.96	79.41	2.21×	2.83	4.16
$h = 3$	123.37	1341.91	10.88×	7.88	75.81	36.00	79.38	2.21×	2.86	4.21
$h = 4$	197.35	2673.21	13.55×	15.26	149.46	36.49	79.39	2.18×	2.90	4.21
$h = 5$	379.69	–	–	34.72	–	36.68	80.05	2.18×	3.00	4.22
$M = 512$	98.35	884.60	8.99×	6.50	39.94	35.96	79.41	2.21×	2.83	4.16
$M = 1024$	185.95	1471.08	7.91×	12.58	76.00	75.66	176.20	2.33×	5.75	8.33
$M = 2048$	378.49	2676.95	7.07×	27.55	148.34	160.78	395.32	2.46×	11.84	16.69
$M = 4096$	943.21	–	–	69.08	–	339.77	848.49	2.50×	25.65	33.41

126.56 s at $T = 4$, versus 211.07 s to 1030.83 s for CertXGB. The resulting speedups are 6.58×–8.15× for training and 1.96×–3.75× for inference; Optdigit* gives 8.99×–9.50× and 2.21×–2.53×, respectively.

Scaling with tree depth. Depth separates the training strategies more strongly: on Dry Bean, the training speedup rises from 6.58× at $h = 2$ to 12.26× at $h = 5$. This occurs because CertXGB pads every binary tree to its full shape, so increasing the maximum depth h essentially doubles the committed forest, even when stopping produces shallower trees. By contrast, TERRAE commits only real nodes and pays more only when their count crosses the next power-of-two boundary; increasing h therefore need not double its committed forest.

The Optdigit* training speedup reaches 13.55× before CertXGB exhausts memory at $h = 5$. Inference on Dry Bean remains nearly flat for both systems:

Table 7. Histogram aggregation microbenchmark when scaling the update length n with fixed address space $m = 2^{15}$.

$\log_2 n$	Prover time (s)			Verifier time (s)			Proof size (KiB)		
	Ours	WLogUp	RAM	Ours	WLogUp	RAM	Ours	WLogUp	RAM
15	5.984	2.120	94.440	0.024	0.005	0.049	1.45	0.70	2.71
16	9.263	3.569	91.627	0.027	0.006	0.053	1.45	0.70	2.71
17	17.026	6.035	193.750	0.028	0.005	0.055	1.45	0.70	2.71
18	31.923	10.859	410.415	0.028	0.006	0.055	1.45	0.70	2.71
19	64.057	21.030	777.688	0.027	0.005	0.055	1.45	0.70	2.71

TERRAE takes 9.78–10.42 s, while CertXGB remains around 36 s because its padded domain does not cross its next boundary.

Scaling with the number of samples. As M increases from 512 to 4096 on Dry Bean, the training speedup decreases from 6.58× to 3.18×, while the inference speedup remains 3.75×–4.31×. Optdigit* shows the same trend, with 7.07×–8.99× training speedups through $M = 2048$ before CertXGB exhausts memory at $M = 4096$.

Scaling with the feature dimension. At the common settings, Optdigit* has four times as many features as Dry Bean. Training time increases by 3.06× for TERRAE and 4.19× for CertXGB; inference time increases by 3.67× and 2.16×. The deviations from exactly 4× reflect feature-independent work and, for CertXGB, power-of-two circuit padding. TERRAE remains 8.99× faster for training and 2.21× faster for inference on the wider dataset.

6.4 Histogram Aggregation Microbenchmark

We isolate the histogram relation in Equation (4.13), comparing TERRAE with ZKBoost v3's weighted-LogUp (WLogUp) [25] and a historical RAM-style construction. Here n is the number of updates and m is the number of histogram buckets. Tables 7 and 8 vary n and m, respectively.

Weighted-LogUp is 2.60×–3.07× faster than TERRAE in prover time, with 0.005–0.006 s verification and 0.70 KiB proofs, compared with 0.024–0.028 s and 1.45 KiB for TERRAE. Both specialized arguments are faster than the RAM baseline throughout. Thus, the full-system speedups do not imply a faster histogram primitive.

Limitations. All measurements are single-threaded and were obtained on one 188 GiB machine. These experiments do not demonstrate production-scale training. Runs approaching 180 GiB triggered an out-of-memory error, so the dashes mark a machine-memory boundary rather than a protocol impossibility. Larger deployments will require parallelism, memory-efficient proving, or distributed execution.

Table 8. Histogram aggregation microbenchmark when scaling the address space m with fixed update length $n = 2^{17}$.

$\log_2 m$	Prover time (s)			Verifier time (s)			Proof size (KiB)		
	Ours	WLogUp	RAM	Ours	WLogUp	RAM	Ours	WLogUp	RAM
14	16.283	5.405	196.333	0.028	0.006	0.053	1.45	0.70	2.71
15	17.026	6.035	193.750	0.028	0.005	0.055	1.45	0.70	2.71
16	19.508	6.721	189.402	0.027	0.006	0.065	1.45	0.70	2.71
17	23.736	7.724	422.247	0.024	0.005	0.052	1.45	0.70	2.71
18	41.002	13.481	778.226	0.024	0.006	0.056	1.45	0.70	2.71

7 Conclusion

Terrae proves quantized GBDT training and inference through structured tensor commitments and specialized arguments for optimal split selection. Domain-lifting and interleaving batching combine algebraic and non-native checks across domains without extra auxiliary commitments. It also utilizes a specialized histogram argument for sample-to-bucket updates, avoiding generic RAM proofs. Together, these techniques substantially reduce prover cost relative to CertXGB [25] under the same KZG backend, at the cost of larger proofs and greater verifier work. This highlights the benefits of exploiting tree structure directly.

8 Open Science

Our source code is publicly available at `https://github.com/GJCPP/Terrae`. The scripts automatically download the public Dry Bean and Optdigits datasets used in our evaluation.

9 Ethical Considerations

Our experiments use public benchmark datasets and do not recruit human participants or collect new personal data. The proofs certify computation on committed inputs; they do not establish data provenance, model fairness, or suitability for high-stakes decisions. Deployment therefore still requires appropriate data governance and assessment of model bias and misuse.

Acknowledgments

The work at Nanjing University was supported by the Natural Science Foundation on Frontier Leading Technology Basic Research Project of Jiangsu (No. BK20222001), NSFC-62272215, AIQ funding of Nanjing University, and the 111 Center (No. B26023).

The work at the National University of Singapore was supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (Award MOE-T2EP20125-0015), the National Research Foundation, Singapore, under the NRF Fellowship, and the National Research Foundation, Singapore, under its CyberSG R&D Programme (CRPO 25-2752-A0002).

References

1. Abbaszadeh, K., Pappas, C., Katz, J., Papadopoulos, D.: Zero-knowledge proofs of training for deep neural networks. In: Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security. pp. 4316–4330 (2024)
2. Alpaydin, E., Kaynak, C.: Optical Recognition of Handwritten Digits. UCI Machine Learning Repository (1998), DOI: https://doi.org/10.24432/C50P49
3. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for r1cs. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019. pp. 103–128. Springer International Publishing, Cham (2019)
4. Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Hirt, M., Smith, A. (eds.) Theory of Cryptography. pp. 31–60. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
5. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
6. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth (1984)
7. Campanelli, M., Faonio, A., Fiore, D., Li, T., Lipmaa, H.: Lookup arguments: Improvements, extensions and applications to zero-knowledge decision trees. In: Tang, Q., Teague, V. (eds.) Public-Key Cryptography – PKC 2024. pp. 337–369. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-57722-2_11, https://eprint.iacr.org/2023/1518
8. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 785–794 (2016). https://doi.org/10.1145/2939672.2939785
9. Choudhuri, A.R., Garg, S., Goel, A., Sekar, S., Sinha, R.: Sublonk: Sublinear prover plonk. Proceedings on Privacy Enhancing Technologies (2024)
10. Danezis, G., Kohlweiss, M., Livshits, B., Rial, A.: Private client-side profiling with random forests and hidden markov models. In: Fischer-Hübner, S., Wright, M. (eds.) Privacy Enhancing Technologies. pp. 18–37. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
11. Eagen, L., Fiore, D., Gabizon, A.: cq: Cached quotients for fast lookups. Cryptology ePrint Archive, Paper 2022/1763 (2022), https://eprint.iacr.org/2022/1763
12. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018. pp. 33–62. Springer International Publishing, Cham (2018)
13. Gabizon, A., Khovratovich, D.: flookup: Fractional decomposition-based lookups in quasi-linear time independent of table size. Cryptology ePrint Archive (2022)
14. Gabizon, A., Williamson, Z.J.: plookup: A simplified polynomial protocol for lookup tables. Cryptology ePrint Archive (2020)
15. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Paper 2019/953 (2019), https://eprint.iacr.org/2019/953

16. Gao, J., Qu, W., Zhang, Y., Zhong, S., Zhang, J.: Zero-knowledge proofs for gradient boosted decision trees. In: Proceedings of the 2026 ACM SIGSAC Conference on Computer and Communications Security (2026). https://doi.org/10.1145/3830454.3846711, https://doi.org/10.1145/3830454.3846711
17. Gorishniy, Y., Rubachev, I., Khrulkov, V., Babenko, A.: Revisiting deep learning models for tabular data. Advances in neural information processing systems **34**, 18932–18943 (2021)
18. Grinsztajn, L., Oyallon, E., Varoquaux, G.: Why do tree-based models still outperform deep learning on typical tabular data? In: Proceedings of the 36th International Conference on Neural Information Processing Systems. NIPS '22, Curran Associates Inc., Red Hook, NY, USA (2022)
19. Haböck, U.: Multivariate lookups based on logarithmic derivatives. Cryptology ePrint Archive (2022)
20. Jia, H., Yaghini, M., Choquette-Choo, C.A., Dullerud, N., Thudi, A., Chandrasekaran, V., Papernot, N.: Proof-of-learning: Definitions and practice. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 1039–1056 (2021). https://doi.org/10.1109/SP40001.2021.00106
21. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) Advances in Cryptology - ASIACRYPT 2010. pp. 177–194. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
22. Koklu, M., Ozkan, I.A.: Multiclass classification of dry beans using computer vision and machine learning techniques. Computers and Electronics in Agriculture **174**, 105507 (2020)
23. Lima, G., Grgić-Hlača, N., Jeong, J.K., Cha, M.: The conflict between explainable and accountable decision-making algorithms. In: Proceedings of the 2022 ACM conference on fairness, accountability, and transparency. pp. 2103–2113 (2022)
24. Masserova, E., Kulkarni, K., Polychroniadou, A., Rothblum, R.D., Takahashi, A.: Audifair: Privacy-preserving framework for auditing fairness. In: Forty-Second Annual Conference on Uncertainty in Artificial Intelligence (2026), https://openreview.net/forum?id=g35Nf6iTlo
25. Melissaris, N., Polychroniadou, A., Takahashi, A., Weng, C., Xu, J.: Zkboost: Zero-knowledge verifiable training for xgboost (2026), https://arxiv.org/abs/2602.04113v3, version 3, 12 May 2026
26. Melissaris, N., Xu, J., Polychroniadou, A., Takahashi, A., Weng, C.: Zkboost: Zero-knowledge verifiable training for xgboost (2026), https://arxiv.org/abs/2602.04113v2, version 2, 9 Feb 2026
27. Pappas, C., Papadopoulos, D.: Sparrow: Space-efficient zksnark for data-parallel circuits and applications to zero-knowledge decision trees. In: Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security. pp. 3110–3124 (2024). https://doi.org/10.1145/3658644.3690318
28. Peng, Z., Zhao, C., Wang, T., Liao, G., Lin, Z., Liu, Y., Cao, B., Shi, L., Yang, Q., Zhang, S.: A survey of zero-knowledge proof based verifiable machine learning. Artificial Intelligence Review (Apr 2026). https://doi.org/10.1007/s10462-026-11557-y, https://doi.org/10.1007/s10462-026-11557-y
29. Posen, J., Kattis, A.A.: Caulk+: Table-independent lookup arguments. Cryptology ePrint Archive (2022)
30. Ribeiro, M., Grolinger, K., Capretz, M.A.: Mlaas: Machine learning as a service. In: 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA). pp. 896–902 (2015). https://doi.org/10.1109/ICMLA.2015.152
31. Setty, S., Lee, J.: Quarks: Quadruple-efficient transparent zkSNARKs (2020), https://eprint.iacr.org/2020/1275

32. Shahu, A., Greinecker, N., Wintersberger, P.: Faster but not smarter? temporal constraints and user compliance with ai explanations. In: Proceedings of the Extended Abstracts of the 2026 CHI Conference on Human Factors in Computing Systems. pp. 1–7 (2026)
33. Shamsabadi, A.S., Wyllie, S.C., Franzese, N., Dullerud, N., Gambs, S., Papernot, N., Wang, X., Weller, A.: Confidential-profitt: Confidential proof of fair training of trees. In: The Eleventh International Conference on Learning Representations (2022)
34. Shigeo, M.: mcl: A portable and fast pairing-based cryptography library. https://github.com/herumi/mcl
35. Shwartz-Ziv, R., Armon, A.: Tabular data: Deep learning is not all you need. Information Fusion **81**, 84–90 (2022). https://doi.org/https://doi.org/10.1016/j.inffus.2021.11.011, https://www.sciencedirect.com/science/article/pii/S1566253521002360
36. Singh, N., Dayama, P., Pandit, V.: Zero knowledge proofs towards verifiable decentralized ai pipelines. In: Eyal, I., Garay, J. (eds.) Financial Cryptography and Data Security. pp. 248–275. Springer International Publishing, Cham (2022)
37. Tanuwidjaja, H.C., Choi, R., Baek, S., Kim, K.: Privacy-preserving deep learning on machine learning as a service—a comprehensive survey. Ieee Access **8**, 167425–167447 (2020)
38. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction apis. In: Proceedings of the 25th USENIX Conference on Security Symposium. p. 601–618. SEC'16, USENIX Association, USA (2016)
39. Zapico, A., Buterin, V., Khovratovich, D., Maller, M., Nitulescu, A., Simkin, M.: Caulk: Lookup arguments in sublinear time. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 3121–3134 (2022)
40. Zapico, A., Gabizon, A., Khovratovich, D., Maller, M., Ràfols, C.: Baloo: Algebraic lookup arguments. IACR Communications in Cryptology **2**(2) (2025). https://doi.org/10.62056/ae89015vt
41. Zhang, J., Fang, Z., Zhang, Y., Song, D.: Zero knowledge proofs for decision tree predictions and accuracy. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. p. 2039–2053. CCS '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3372297.3417278, https://doi.org/10.1145/3372297.3417278
42. Zhang, Q., Xiang, T., Cai, Y., Zhao, Z., Wang, N., Wu, H.: Privacy-preserving machine learning as a service: Challenges and opportunities. IEEE Network **37**(6), 214–223 (2023). https://doi.org/10.1109/MNET.127.2200342
43. Zhang, Y., Sun, S.F., Gu, D.: Efficient kzg-based univariate sum-check and lookup argument. In: IACR International Conference on Public-Key Cryptography. pp. 400–425. Springer (2024)
44. Zhao, L., Wang, Q., Wang, C., Li, Q., Shen, C., Feng, B.: 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). https://doi.org/10.1109/TPDS.2021.3068195

A Generative AI Usage

The authors used generative AI tools only for language-level assistance, such as grammar correction, spelling correction, phrasing suggestions, and improving the readability of existing text. The protocol design, mathematical formulation, security and soundness analysis, implementation, experiments, interpretation of results, and all technical claims were developed, checked, and approved by the authors. The authors take full responsibility for the content of the paper.

B Related Work

B.1 Zero-Knowledge Proofs for Tree-Based Models

Zero-knowledge proofs for tree-based machine learning models have been studied under several different goals: proving inference, proving accuracy or fairness properties, and proving training. Early work on private client-side profiling already considered random forests together with zero-knowledge proofs of correct model evaluation [10]. Later, Zhang et al. [41] initiated the systematic study of zero-knowledge decision trees. Their zkDT protocols allow a prover to prove the correctness of a decision-tree prediction or an accuracy claim without revealing the model. They also extend the approach to random forests by proving the predictions of individual trees and aggregating their outputs. This line of work is primarily concerned with inference and accuracy testing for already trained models, rather than proving that a model is the output of a prescribed training algorithm.

Several subsequent works improve or extend zero-knowledge proofs for tree inference. Singh et al. [36] study verifiable decentralized AI pipelines and include protocols for verifiable inference of decision trees and random forests. Campanelli et al. [7] introduce lookup argument techniques, including matrix lookup, and apply them to zero-knowledge decision trees. They use a hyper-box encoding of a decision tree, which, composed with a table-independent lookup argument [7, 11], allows tree-independent inference proofs for decision trees.

Another related direction is zero-knowledge auditing of fairness for tree-based models. Confidential-PROFITT [33] proposes fair decision tree learning algorithms together with customized zero-knowledge protocols that certify fairness while keeping both the model and training data confidential; it also extends the approach to ensembles of trees. Recent auditing frameworks further study robustness of fairness certificates and transparent generation of audit data [24]. These works prove statistical or fairness properties of a committed model, or prove that a model was produced by a fairness-aware training rule. In contrast, our work targets the full training semantics of gradient-boosted decision trees, including the algebraic constraints for histogram construction, split selection, leaf-value computation, and boosting state updates.

The closest works to ours are zero-knowledge proofs of training for forest-based models. Pappas and Papadopoulos [27] propose a space-efficient zkSNARK, Sparrow, for data-parallel circuits and apply it to zero-knowledge proofs of random

forest training and prediction. More recently, [25] presents a zero-knowledge proof of training for XGBoost. They reformulate XGBoost as a deterministic fixed-point algorithm and give a generic circuit strategy referred to as CertXGB. [25] also gives a concrete construction, ZKBoost, by instantiating CertXGB with a VOLE-based backend. Our work is complementary but takes a different approach: rather than expressing the whole trainer as a general arithmetic circuit, we design specialized algebraic protocols for the structure of GBDT training. In particular, we exploit the histogram-based nature of tree construction and batch the repeated linear, lookup, and aggregation relations that arise across nodes, features, bins, classes, and boosting rounds.

Overall, prior work on zero-knowledge tree models has mainly focused on verifiable inference, accuracy/fairness auditing, or general-purpose circuit encodings of tree training. Our main contribution is a specialized proof system, Terrae, for GBDT training that directly captures the committed training trace and proves that the committed ensemble is obtained by executing the public training algorithm on the committed dataset. Terrae also supports proof of inference from the same committed forest.

B.2 Batching Lookup Arguments

As a corollary of Lemma D.5, we can batch multiple lookup arguments into one, even if (1) the contents are of different sizes, and (2) the tables are different and potentially of different sizes. To be specific, given KZG commitments to content vectors $\{\mathbf{f}_i \in \mathbb{F}^{m_i}\}_{i\in[n]}$ and table vectors $\{\mathbf{t}_i \in \mathbb{F}^{M_i}\}_{i\in[n]}$, we want to prove that, *pairwise*,

$$\mathbf{f}_i \subseteq \mathbf{t}_i \quad \forall i \in [n],$$

where "$\subseteq$" denotes that every element in $\mathbf{f}_i$ is also in $\mathbf{t}_i$.

To the best of our knowledge, this type of batching is not studied in the literature. Existing works on lookup arguments typically fall into one of the following categories:

Batching many lookups into a common table. A common form of batching is to prove that many queried values, or many queried columns, belong to the same lookup table. This is the setting handled by standard Plonkish lookup arguments such as Plookup [14], as well as logarithmic-derivative-based lookup arguments such as LogUp [19]. These protocols amortize the cost of many membership checks by reducing them to one multiset relation against a single table. When multiple tables are supported, the usual approach is to concatenate the tables and add a table identifier, or tag, so that a lookup into table i is encoded as a lookup of the tuple (i, x) into one global tagged table.

This technique is effective when all lookups can be represented as membership in *a single combined table*, but it does not directly batch a family of *independent* committed relations

$$\mathbf{f}_i \subseteq \mathbf{t}_i \quad \forall i \in [n],$$

where both the queried vectors and the corresponding tables are separately committed and may have unrelated sizes. We note that committing contents and tables separately is helpful in our GBDT construction, as it allows us to fetch different forest attributes and process them separately.

Large-table lookup arguments. Another line of work focuses on reducing the cost of lookup arguments when the table is large. Examples include Caulk, Caulk+, flookup, Baloo, cq, and subsequent improvements [11, 13, 29, 39, 40, 43]. The main goal of this line is to make the prover sublinear, table-independent, or quasi-linear in the number of lookups after preprocessing. This direction is orthogonal to our batching problem.

Vector, matrix, and segment lookup arguments. Recent works also study richer lookup relations, such as vector lookup, matrix lookup, and segment lookup [7, 9]. These arguments can prove that a committed vector, row, subvector, or submatrix is contained in a larger committed table, often by adding positional labels or by reducing the relation to a standard lookup. In our construction, we use the matrix lookup of [7], which is nevertheless orthogonal to the batching problem.

In summary, prior batching techniques typically batch many queries into one table, encode multiple tables using tags in one global table, or prove richer containment relations with respect to one committed table-like object. By contrast, our interleaving lemma gives a direct way to batch a family of independent committed lookup relations $\{(\mathbf{f}_i, \mathbf{t}_i)\}_i$ into one single lookup argument, while allowing both the query vectors and the tables to have different sizes.

C Detailed Evaluation Results

C.1 Full-System Performance and Structural Workloads

Implementation corrections. Relative to the original artifact, TERRAE's implementation now certifies the exact candidate-validity mask and Conditions 4, 10, and 15 in Section 5.3: no ghost nodes, no children for leaves, and zero scores on internal nodes. It also certifies the complete inference relation. In particular, training now includes forest well-formedness for both systems, while inference includes routing, score aggregation, and argmax and assumes a previously accepted well-formedness proof for the same forest. We also corrected the original CertXGB implementation and applied several engineering optimizations to both systems, including binary proof serialization. Hence all reported results use this complete proof scope.

Table 9 records the artifact evolution. The original-artifact columns use the original trainer configuration and depth convention, whereas the correctness-only and final columns use the revised minimal Dry Bean setting and isolate the effect of the final engineering optimizations. The columns document artifact evolution rather than a controlled ablation. Under the revised minimal case, the final optimizations reduce TERRAE's training/inference prover time from 61.20/16.80 s to 32.08/9.81 s and CertXGB's from 420.6/152.71 s to 211.07/36.73 s; the

Table 9. Effect of implementation corrections on Dry Bean. The original-artifact columns reproduce the measurements reported under the original trainer configuration and depth convention and are included only to document artifact evolution. The correctness-only and final columns use the revised minimal setting ($M = b = 512$, $T = 1$, $h = 2$, $B = 8$, $\mu = 4$, and $\tau_{\text{split}} = 1/4$). The correctness-only implementation includes the corrected proof scope before the final engineering optimizations.

System	Stage	Original artifact				Correctness-only				Final implementation			
		P (s)	V (s)	M (GiB)	Size (KiB)	P (s)	V (s)	M (GiB)	Size (KiB)	P (s)	V (s)	M (GiB)	Size (KiB)
TERRAE	Training	58.79	0.199	3.79	70.8	61.20	0.139	2.77	72.4	32.08	0.387	2.19	29.81
TERRAE	Inference	15.76	0.062	1.25	3.7	16.80	0.073	1.15	6.3	9.81	0.067	0.81	4.19
CertXGB	Training	288.00	0.026	10.39	12.5	420.6	1.156	21.94	14.7	211.07	0.065	10.40	7.22
CertXGB	Inference	86.94	0.0069	4.16	2.6	152.71	0.0116	5.71	2.6	36.73	0.008	2.19	1.15

The original proofs used ASCII serialization; the correctness-only measurements use partial Terrae optimizations and the original CertXGB baseline optimizations; and the final proofs use raw binary serialization. P, V, and M denote prover time, verifier time, and peak memory, respectively.

corresponding verifier-time, peak-memory, and proof-size changes are reported in the table.

Structural and PCS workloads. Tables 10 and 11 give the structural workloads for every scaling experiment, including CertXGB's constraints and both systems' PCS workloads, which expose remaining implementation effects. TERRAE has no directly comparable compiled-gate count. The "Coeffs." column reports the number of polynomial coefficients processed by PCS, which roughly reflects the amount of data committed. Across completed settings, TERRAE processes 8.35×–19.54× fewer training and 3.65×–7.19× fewer inference coefficients than CertXGB. For TERRAE, the counts remain near 226–232 training and 25 inference commitments, 477–499 training and 66–68 inference claims, and 78–90 training and 22–24 inference pairings. These counts often exceed CertXGB's, yet TERRAE remains faster. Thus, polynomial length, rather than the number of PCS objects, better explains prover time and memory. Batching leaves one final KZG opening proof per completed stage; Table 4 shows the corresponding proof-size and verification tradeoff.

The workloads also explain the scaling trends in Tables 5 and 6. Increasing T or M increases the coefficient volume approximately linearly. Increasing h grows CertXGB's full-tree circuit more rapidly, while TERRAE commits only real nodes. During inference, TERRAE's workload is constant across the tested depths. CertXGB's constraint count grows with h, but its coefficient count and prover time remain flat until the circuit crosses the next power-of-two PLONK domain. At the common settings, Optdigit*'s fourfold feature dimension raises training coefficients by 3.32× for TERRAE and 3.44× for CertXGB, showing that feature-dependent histogram work dominates the cross-dataset difference.

On Dry Bean, CertXGB crosses a power-of-two PLONK-domain boundary in inference at $T = 4$, causing its inference coefficient count to double and its inference prover time to increase accordingly. Its training coefficient count instead grows linearly with T. As sample-dependent work dominates from $M = 512$ to

Table 10. Structural and PCS workloads on Dry Bean. Paired entries report TERRAE/CertXGB. Coeffs. count polynomial coefficients processed by all KZG commitments, including input-independent preprocessing; PCSs count committed polynomials; Forest nodes count active split and leaf nodes in the shared forest; opening claims count polynomial–point evaluation claims before batching; pairings are verifier operations; constraints are CertXGB's compiled active PLONK gates. Training includes forest well-formedness.

Setting	Nodes	Training						Inference					
		TERRAE/CertXGB					CertXGB	TERRAE/CertXGB					CertXGB
		Coeffs. (M)		PCSs	Claims	Pairings	Constr. (K)	Coeffs. (M)		PCSs	Claims	Pairings	Constr. (K)
$T = 1$	21	2.73	34.05	229 170	488 64	82 30	1116.34	0.84	6.03	25 27 66 33		22 15	133.17
$T = 2$	42	5.42	68.10	229 170	499 96	82 30	2217.02	1.65	6.03	25 27 68 65		24 15	178.26
$T = 4$	84	10.69	136.21	229 170	497 160	86 30	4418.39	3.27	12.06	25 27 68 129		24 15	268.43
$h = 2$	21	2.73	34.05	229 170	488 64	82 30	1116.34	0.84	6.03	25 27 66 33		22 15	133.17
$h = 3$	45	3.23	49.75	229 170	488 106	79 29	1713.41	0.84	6.03	25 27 66 75		22 14	149.58
$h = 4$	87	4.50	86.66	229 170	489 181	79 29	2868.38	0.84	6.03	25 27 66 150		22 14	165.99
$h = 5$	139	7.03	137.41	229 170	488 303	79 29	5129.14	0.84	6.03	25 27 66 272		22 14	182.40
$M = 512$	21	2.73	34.05	229 170	488 64	82 30	1116.34	0.84	6.03	25 27 66 33		22 15	133.17
$M = 1024$	21	5.68	55.42	230 170	489 64	86 30	1676.47	1.68	12.06	25 27 66 33		23 15	266.29
$M = 2048$	21	11.22	98.15	227 170	483 64	90 30	2821.30	3.35	24.12	25 27 66 33		22 15	532.53
$M = 4096$	21	21.99	183.60	228 170	486 64	90 30	5061.81	6.71	48.23	25 27 66 33		22 15	1065.01

4096, the training coefficient ratio narrows from 12.48× to 8.35× and the training speedup from 6.58× to 3.18×. The inference coefficient ratio stays about 7.2×, matching the stable 3.75×–4.31× speedup.

Practical performance. Tables 12 and 13 connect these structural trends to heavier and more practical settings, including held-out accuracy. These experiments fix $M = b = 1024$, $h = 3$, and $B = 8$ and scale T.

Accuracy improves with T, rising from 84.63% to 88.93% on Dry Bean and from 77.85% to 92.41% on Optdigit*. Over completed comparisons, TERRAE is 6.16×–9.06× faster for training and 2.18×–4.12× faster for inference. The coefficient counts grow nearly linearly with T and track both prover time and peak memory. This also explains the OOM-terminated runs: CertXGB reaches 163.88 GiB at Dry Bean $T = 8$ and 157.56 GiB at Optdigit* $T = 2$, so its larger Optdigit* training settings exceed the machine-memory budget. TERRAE completes Dry Bean $T = 8$ and Optdigit* $T = 4$; only its Optdigit* $T = 8$ training run exhausts memory.

D Proof-Batching Tools

D.1 Domain-Lifting Batching

Most algebraic constraints in our construction are expressed as polynomial identities over power-of-two evaluation domains. The following two theorems lift claims over different domains to a common domain and then batch claims of the same form. The lifting polynomial $\nu_N(X)/\nu_{n_i}(X)$ implicitly appears in the

Table 11. Structural and PCS workloads on Optdigit*. Forest nodes count active split and leaf nodes in the shared forest. Metrics follow Table 10; Coeffs. include input-independent preprocessing. A dash denotes unavailable data due to memory limit.

Setting	Nodes	Training TERRAE/CertXGB								CertXGB	Inference TERRAE/CertXGB								CertXGB
		Coeffs.	(M)	PCSs		Claims		Pairings		Constr. (K)	Coeffs.	(M)	PCSs		Claims		Pairings		Constr. (K)
$T = 1$	21	9.07	117.16	227	170	481	64	81	30	3782.07	3.20	12.06	25	27	66	33	23	15	133.18
$T = 2$	42	18.11	234.31	230	170	497	96	86	30	7526.21	6.37	24.12	25	27	68	65	24	15	178.27
$T = 4$	84	36.06	468.62	230	170	495	160	85	30	15014.49	12.70	48.23	25	27	68	129	24	15	268.46
$h = 2$	21	9.07	117.16	227	170	481	64	81	30	3782.07	3.20	12.06	25	27	66	33	23	15	133.18
$h = 3$	49	11.09	167.73	230	170	487	108	78	29	6054.92	3.20	12.06	25	27	66	77	22	14	149.58
$h = 4$	89	16.18	315.01	230	170	488	186	80	29	10539.16	3.20	12.06	25	27	66	155	22	14	165.99
$h = 5$	121	26.37	–	230	–	487	–	79	–	–	3.20	12.06	25	27	66	269	22	14	182.40
$M = 512$	21	9.07	117.16	227	170	481	64	81	30	3782.07	3.20	12.06	25	27	66	33	23	15	133.18
$M = 1024$	21	16.39	183.74	232	170	491	64	79	30	5349.82	6.40	24.12	25	27	66	33	23	15	266.30
$M = 2048$	21	30.46	316.91	232	170	491	64	86	30	8583.61	12.79	48.23	25	27	66	33	22	15	532.54
$M = 4096$	21	62.41	–	226	–	477	–	89	–	–	25.58	96.47	25	27	66	33	23	15	1065.02

Table 12. Practical-setting performance at $M = b = 1024$, $h = 3$, and $B = 8$. Accuracy is evaluated on the held-out sets of 2,720 Dry Bean and 1,264 Optdigit* samples. For both systems, training includes forest well-formedness. A dash denotes an OOM-limited training setting.

Dataset	T	Acc. (%)	Training Time (s)			Training Memory (GiB)		Inference Time (s)			Inference Memory (GiB)	
			TERRAE	CertXGB	Speedup	TERRAE	CertXGB	TERRAE	CertXGB	Speedup	TERRAE	CertXGB
Dry Bean	1	84.63	78.29	495.59	6.33×	6.18	20.83	19.84	81.79	4.12×	1.72	4.21
	2	86.07	162.40	1112.15	6.85×	13.15	41.45	37.68	82.11	2.18×	3.08	4.30
	4	88.09	342.97	2419.44	7.05×	29.24	82.83	75.11	176.89	2.35×	5.80	8.45
	8	88.93	754.43	4645.83	6.16×	70.82	163.88	160.27	395.94	2.47×	11.21	16.92
Optdigit*	1	77.85	212.13	1922.70	9.06×	14.57	79.54	75.24	178.68	2.37×	5.77	8.34
	2	84.18	454.61	4068.53	8.95×	31.61	157.56	157.30	398.49	2.53×	11.12	16.71
	4	90.11	996.92	–	–	73.01	–	327.29	759.84	2.32×	21.92	33.32
	8	92.41	–	–	–	–	–	669.39	1752.99	2.62×	43.43	66.52

CQ++ lookup argument [7]; here we isolate its use for general zero and sum claims. Recall that as the KZG SRS supports commitments only up to a degree bound $d_{\max}$, all committed quotient and remainder polynomials below must lie within this bound and hence are low-degree; Appendix I verifies these degree requirements.

Theorem D.1 (Domain-lifting batching for zero checks). *Let $2 \le n_0 \le \cdots \le n_{m-1} = N$ be powers of two, and let the ith claim be that f_i vanishes on $\mathbb{H}_{n_i}$. Define*

$$\widehat{f}_i(X) := f_i(X)\frac{\nu_N(X)}{\nu_{n_i}(X)}.$$

After the polynomials f_i have been fixed, the verifier samples η uniformly from $\mathbb{F}$ and checks that

$$\sum_{i\in[m]} \eta^i \widehat{f}_i(X) = q(X)\nu_N(X)$$

Table 13. Structural and PCS workloads in the practical setting ($M = b = 1024$, $h = 3$, $B = 8$). Forest nodes count active split and leaf nodes for the seven real classes in the shared forest. Paired entries report TERRAE/CertXGB. Coeffs. count polynomial coefficients processed by all KZG commitments, including preprocessing; PCSs count committed polynomials; Claims are polynomial–point claims before batching; pairings are verifier operations. A dash denotes that the run triggers an out-of-memory error.

Dataset	T	Nodes	Training								Inference							
			TERRAE/CertXGB							CertXGB	TERRAE/CertXGB							CertXGB
			Coeffs. (M)		PCSs		Claims		Pairings	Constr. (K)	Coeffs. (M)		PCSs		Claims		Pairings	Constr. (K)
Dry Bean	1	45	6.28	74.13	230	170	490	106	83 29	2310.40	1.68	12.06	25	27	66	75	23 14	299.08
Dry Bean	2	94	12.64	148.27	233	170	503	185	82 29	4591.07	3.30	12.06	25	27	68	154	25 14	422.02
Dry Bean	4	192	25.09	296.53	230	170	498	343	87 29	9152.40	6.53	24.12	25	27	68	312	25 14	667.88
Dry Bean	8	382	50.03	593.06	230	170	499	652	86 29	18533.36	12.98	48.24	25	27	68	621	24 13	1159.54
Optdigit*	1	49	18.80	234.31	235	170	498	108	81 29	7659.53	6.40	24.12	25	27	66	77	23 14	299.09
Optdigit*	2	98	37.66	468.62	236	170	507	188	85 29	15246.31	12.74	48.23	25	27	68	157	25 14	422.04
Optdigit*	4	196	75.05	–	235	–	506	–	85 –	–	25.40	96.47	25	27	68	317	25 13	667.92
Optdigit*	8	392	–	–	–	–	–	–	– –	–	50.74	192.94	25	27	68	633	23 14	1159.60

for a low-degree quotient polynomial q. If any original zero claim is false, the batched claim holds with probability at most $(m-1)/p$ over η.

Proof. Let $a_i(X) := f_i(X) \bmod \nu_{n_i}(X)$. The ith claim is true exactly when $a_i = 0$. Its lifted residual

$$\widehat{a}_i(X) := a_i(X)\frac{\nu_N(X)}{\nu_{n_i}(X)}$$

has degree smaller than N and is nonzero whenever a_i is nonzero. Consequently,

$$A_\eta(X) := \sum_{i\in[m]} \eta^i \widehat{a}_i(X)$$

is the zero polynomial only if every coefficient of A_η vanishes. If some a_i is nonzero, at least one coefficient of A_η is a nonzero polynomial in η of degree at most $m-1$. Hence $A_\eta = 0$ for at most $m-1$ values of η. Additionally, unless $A_\eta = 0$, $A_\eta = q(X)\nu_N(X)$ cannot be true for a low-degree polynomial q as A_η has degree smaller than N. Thus we bound the probability by $(m-1)/p$.

Theorem D.2 (Domain-lifting batching for sum checks). *Let $2 \le n_0 \le \cdots \le n_{m-1} = N$ be powers of two, and let the ith claim be*

$$\sum_{x\in\mathbb{H}_{n_i}} f_i(x) = s_i.$$

Set $g_i(X) := f_i(X) - s_i/n_i$ and

$$\widehat{g}_i(X) := g_i(X)\frac{\nu_N(X)}{\nu_{n_i}(X)}.$$

After the polynomials f_i have been fixed, the verifier samples η uniformly from $\mathbb{F}$ and checks

$$\deg r(X) < N - 1, \tag{D.1}$$

$$\sum_{i\in[m]} \eta^i \widehat{g}_i(X) = Xr(X) + q(X)\nu_N(X). \tag{D.2}$$

If any original sum claim is false, these conditions hold with probability at most $(m-1)/p$ over η.

Proof. Let $a_i(X) := g_i(X) \bmod \nu_{n_i}(X)$. Since $\deg a_i < n_i$, the sum of g_i over $\mathbb{H}_{n_i}$ equals n_i times the constant coefficient of a_i, by Aurora's sumcheck [3]. Thus the ith sum claim is true exactly when $a_i(0) = 0$, or equivalently when $a_i(X) = Xr_i(X)$ for some $\deg r_i < n_i - 1$.

The constant coefficient of $a_i(X)\nu_N(X)/\nu_{n_i}(X)$ equals $a_i(0)$ because the lifting polynomial has constant coefficient one and the product has degree smaller than N. Therefore, if some sum claim is false, the constant coefficient of

$$\sum_{i\in[m]} \eta^i a_i(X) \frac{\nu_N(X)}{\nu_{n_i}(X)}$$

is a nonzero polynomial in η of degree at most $m-1$. It vanishes with probability at most $(m-1)/p$.

When that constant coefficient is zero, the lifted residual is divisible by X. Its degree is smaller than N, so the corresponding polynomial r has degree smaller than $N-1$. The implementation commits to this batched remainder and proves the bound $\deg r < N - 1$ with the KZG degree-check mechanism.

Zero checks and sum checks use separate batching challenges because their accepted residual spaces are different. Claims within either family may be batched across domains, but the two families are not mixed.

D.2 Interleaving

The intended setting for interleaving batching is that many non-native predicate instances share the same local rule and the same public parameters, but arrive as separate PCS commitments and witnesses on possibly different power-of-two domains. We first record the alignment lemma used to place smaller objects on larger domains.

Lemma D.3 (Row-replication). *Let $\mathbf{v} \in \mathbb{F}^m$ and let $v(X)$ be its encoding over $\mathbb{H}_m$. Let $\mathbf{V} \in \mathbb{F}^{r\times m}$ be the matrix each of whose rows equals $\mathbf{v}$. Then the encoding polynomial of $\mathbf{V}$ over $\mathbb{H}_{rm}$ is*

$$V(X) = v(X^r).$$

Proof. For every $i \in [r]$ and $j \in [m]$,

$$V(\omega_{rm}^{im+j}) = \mathbf{V}[i,j] = \mathbf{v}[j] = v(\omega_m^j) = v((\omega_{rm}^{im+j})^r),$$

where the last equality uses $\omega_{rm}^r = \omega_m$. Since both sides have degree less than rm and agree on $\mathbb{H}_{rm}$, they are identical.

Definition D.4 (Pairwise interleaving). *Let $f(X)$ and $g(X)$ be the encoding polynomials of two length-n vectors over $\mathbb{H}_n$, where n is a power of two. Define*

$$\mathrm{even}_n(X) := \frac{X^n + 1}{2}, \qquad \mathrm{odd}_n(X) := \frac{1 - X^n}{2}.$$

Their evaluations on $\mathbb{H}_{2n}$ are the alternating selectors $(1,0,1,0,\ldots)$ and $(0,1,0,1,\ldots)$. The pairwise interleaving of $f(X)$ and $g(X)$ is defined as the virtual polynomial

$$\mathrm{Int}_n(f,g)(X) := \mathrm{even}_n(X)\, f(X) + \mathrm{odd}_n(X)\, g(\omega_{2n}^{-1} X).$$

Its evaluation vector over $\mathbb{H}_{2n}$ is

$$(f[0], g[0], f[1], g[1], \ldots, f[n-1], g[n-1]).$$

Lemma D.5 (Interleaving). *For each $i \in [m]$, let $u_i(X)$ be a polynomial encoding $\mathbf{u}_i$ over $\mathbb{H}_{n_i}$, where each n_i is a power of two. Let $S = \sum_{i\in[m]} n_i$ and $\widehat{n} = 2^{\lceil \log_2 S \rceil}$. There is a deterministic algorithm that interleaves these objects into a virtual polynomial $\widehat{u}(X)$ over $\mathbb{H}_{\widehat{n}}$ and guarantees:*

1. *the evaluations before and after interleaving have the same underlying set of entries:*
$$\{\widehat{u}(a) : a \in \mathbb{H}_{\widehat{n}}\} = \bigcup_{i\in[m]} \{u_i(a) : a \in \mathbb{H}_{n_i}\};$$
2. *for any $x \in \mathbb{F}$, given oracle access to $\{u_i\}_{i\in[m]}$, $\widehat{u}(x)$ can be evaluated with one query to each u_i, at points deterministically derived from x, and $O(m + \log \widehat{n})$ field operations;*
3. *the algorithm runs in $O(m + \log \widehat{n})$ time*[10].

We denote this algorithm by

$$(\widehat{\mathbf{u}}, \widehat{n}) \leftarrow \mathsf{Interleave}((\mathbf{u}_i, n_i)_{i\in[m]}).$$

For simplicity, we may omit the domain sizes when they are clear from context.

Proof. Start from the active multiset $\{(u_i(X), n_i)\}_{i\in[m]}$. We first store all pairs with the same domain size in the same bucket. Then we perform two scans over the buckets. In the first scan, process the buckets from small to large domain size

[10] This complexity is for constructing the virtual merge-tree representation of $\widehat{u}$ from $\{u_i\}$, without materializing its $\widehat{n}$ coefficients or evaluations.

and merge equal-sized pairs whenever a bucket contains at least two active pairs. Each such merge removes two equal-sized pairs, uses pairwise interleaving as in Definition D.4 to create a new pair, and inserts the new pair into the appropriate bucket. After this pass, each bucket contains at most one active pair.

In the second scan, repeatedly take the two active pairs with smallest domain size $n_i \le n_j$. If $n_i \neq n_j$, replicate the smaller one by mapping $X \mapsto X^{n_j/n_i}$ as stated in Lemma D.3, then interleave the pair. Repeat until one active pair $(\widehat{u}(X), \widehat{n})$ remains.

Because the domain sizes are powers of two, the bucket structure contains at most $\log \widehat{n}$ levels. Initial bucketing costs $O(m)$. Both scans can advance monotonically through these levels, and there are at most $m-1$ merges, each of which creates a constant-size virtual expression. Thus the algorithm runs in $O(m + \log \widehat{n})$ time.

For set preservation, lifting is exactly the setting of Lemma D.3: mapping $X \mapsto X^{n_j/n_i}$ repeats $\mathbf{u}_i$ on the larger domain $\mathbb{H}_{n_j}$ and introduces no new elements. Clearly, interleaving also introduces no new elements. As both operations preserve the elements in the vector, repeating them gives the claimed set equality.

For the final domain size, note that the first scan preserves the total size S and leaves at most one active pair in each bucket. Denote the remaining pairs after the first scan by $(n'_i, u'_i(X))$ with

$$n'_1 < \cdots < n'_k,$$

then

$$\sum_{i=1}^{k} n'_i = S \qquad \text{and} \qquad \log n'_k = \lfloor \log_2 S \rfloor.$$

If $k = 1$, then S is already a power of two and $\widehat{n} = S$. Otherwise the second scan merges the remaining powers upward and can push the final size no higher than $2n'_k$. Since $S > n'_k$, we have

$$\log \widehat{n} = 1 + \log n'_k = \lceil \log_2 S \rceil.$$

This is exactly the least power of two not smaller than S.

For evaluation, record the merge tree and propagate the target point x from its root to its leaves. Pairwise interleaving sends a point z to z and $\omega_{2n}^{-1} z$ for its two children, while row replication sends z to z^r. Thus every leaf receives a point deterministically derived from x, and the evaluation requires one oracle query to each original polynomial. Precompute $x, x^2, x^{2^2}, \ldots, x^{2^{\lceil \log_2 \widehat{n} \rceil}}$ in $O(\log \widehat{n})$ field operations. Every propagated point has the form ζx^{2^e} for a public root-of-unity factor ζ, so each replication, selector evaluation, and merge is then computed in $O(1)$ field operations. The tree has m leaves and $m-1$ internal nodes, giving $O(m + \log \widehat{n})$ field operations in total.

D.3 Interleaving Batching

In our proof system, we use interleaving batching mainly to combine two types of check: lookup argument and fixed-point quantization check.

Batching lookup. We first note that, like [7], all table lookups in our construction are first reduced to vector lookups. It is hence without loss of generality to focus on batching vector lookups.

Suppose we have several lookup families indexed by $i \in [s]$. For each family, there is a public table $\mathbf{t}_i$ and a collection of vector queries $\mathbf{f}_{i,1}, \ldots, \mathbf{f}_{i,m_i}$ such that every query satisfies $\mathbf{f}_{i,j} \subseteq \mathbf{t}_i$.

We batch them in two steps. First, for each fixed i, we batch all queries $\mathbf{f}_{i,j} \subseteq \mathbf{t}_i$ into one query $\mathbf{f}_i \subseteq \mathbf{t}_i$, by interleaving $\mathbf{f}_{i,j}$ with Lemma D.5. Second, after the query and table commitments are fixed, the verifier samples a random challenge α. For each family $i \in [s]$, using distinct field labels for the family indices, define

$$\mathbf{f}'_i := \alpha i + \mathbf{f}_i, \qquad \mathbf{t}'_i := \alpha i + \mathbf{t}_i.$$

Adding the same family tag to both sides preserves $\mathbf{f}_i \subseteq \mathbf{t}_i$. We interleave the shifted queries into one vector $\mathbf{f}$ and the shifted tables into one vector $\mathbf{t}$.

This yields a single lookup claim

$$\mathbf{f} \subseteq \mathbf{t},$$

which can be proved with a standard lookup argument. The benefit is that the prover amortizes the lookup proof across all tables while keeping the query structure explicit. For soundness, suppose some original relation is false, and fix one such family i^* and one query value $x^* \in \mathbf{f}_{i^*}$ that does not occur in $\mathbf{t}_{i^*}$. If the merged lookup nevertheless holds, then $x^* + \alpha i^* = y + \alpha j$ for some $j \neq i^*$ and $y \in \mathbf{t}_j$; a same-family match is impossible by the choice of x^*. Each fixed pair (j, y) determines at most one value of α. Thus, if L bounds each table length,

$$\epsilon_{\mathsf{sep}}(s, L) \leq \frac{\sum_{j \neq i^*} |\mathbf{t}_j|}{p} \leq \frac{(s-1)L}{p}.$$

Hence, if there is at least one incorrect lookup instance, it gets caught with at least $1 - (s-1)L/p$ probability. This bounds the soundness error.

Batching quantizations. Suppose we are given $\mathbf{a}_i \in \mathbb{F}^{n_i}$ and $\mathbf{b}_i \in \mathbb{F}^{n_i}$ for $i \in [m]$. We want to prove that $\mathbf{a}_i$ is a quantization of $\mathbf{b}_i$, i.e.

$$\forall j \in [n_i].\ \mathbf{a}_i[j] = \left\lfloor \frac{\mathbf{b}_i[j]}{2^q} \right\rfloor \tag{D.3}$$

for the same public divisor 2^q. Such constraints can be batched easily by interleaving

$$(\mathbf{a}, n) \leftarrow \mathsf{Interleave}(\mathbf{a}_0, \ldots, \mathbf{a}_{m-1}), \tag{D.4}$$

$$(\mathbf{b}, n) \leftarrow \mathsf{Interleave}(\mathbf{b}_0, \ldots, \mathbf{b}_{m-1}), \tag{D.5}$$

and proving

$$\forall j \in [n].\ \mathbf{a}[j] = \left\lfloor \frac{\mathbf{b}[j]}{2^q} \right\rfloor. \tag{D.6}$$

Note that as interleaving is deterministic, $\mathbf{a}$ is aligned with $\mathbf{b}$, and the batching incurs no extra soundness error.

E ZK-GBDT

As outlined in Section 5.1, we commit the forest by committing

$$\mathcal{F} = (\mathbf{lc}, \mathbf{rc}, \mathbf{par}, \mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow}, \mathbf{E}, \mathbf{tree}, \mathbf{root}, \mathbf{depth}, \mathbf{score}, \mathbf{is_leaf}, \mathbf{valid}). \quad \text{(E.1)}$$

E.1 Proof Primitives

Zero-check. Given $f(X)$ encoding a vector $\mathbf{f}$ over $\mathbb{H}_n$, the zero-check validates that $\mathbf{f} = \mathbf{0}$ by proving the existence of $q(X)$ such that

$$f(X) = q(X) \cdot \nu_n(X). \quad \text{(E.2)}$$

As noted in Theorem D.1, all zero-checks can be batched together.

Sum-check. Given $f(X)$ encoding a vector $\mathbf{f}$ over $\mathbb{H}_n$, the Aurora sum-check [3] validates that $\sum_{i=0}^{n-1} \mathbf{f}[i] = s$ by proving the existence of $q(X)$ and $r(X)$ such that

$$\deg r(X) < n - 1, \quad \text{(E.3)}$$

$$f(X) - s/n = r(X) \cdot X + q(X) \cdot \nu_n(X). \quad \text{(E.4)}$$

As noted in Theorem D.2, all sum-checks can be batched together.

Table lookup. Given $(\mathbf{F}_i)_{i\in[k]}$ and $(\mathbf{T}_i)_{i\in[k]}$ encoding matrices $\mathbf{F}_i \in \mathbb{F}^{n\times m_i}$ and $\mathbf{T}_i \in \mathbb{F}^{N\times m_i}$, the lookup relation validates that there exist $\mathbf{j} \in [N]^n$, such that

$$\mathbf{F}_i[r, c] = \mathbf{T}_i[\mathbf{j}[r], c], \forall r \in [n], c \in [m_i], i \in [k].$$

The concrete construction is deferred to Appendix G.1.

Quantization. Given $f(X)$ and $g(X)$ encoding vectors $\mathbf{f}$ and $\mathbf{g}$ over $\mathbb{H}_n$, and given a public quantization factor q, the quantization relation validates that

$$\mathbf{f}[i] = \left\lfloor \frac{\mathbf{g}[i]}{2^q} \right\rfloor, \forall i \in [n]. \quad \text{(E.5)}$$

This is reduced to proving the existence of $\mathbf{q}$ and $\mathbf{r}$ such that

$$\mathbf{g}[i] = 2^q \cdot \mathbf{f}[i] + \mathbf{r}[i], \forall i \in [n], \quad \text{(E.6)}$$

$$0 \le \mathbf{r}[i] < 2^q, \forall i \in [n]. \quad \text{(E.7)}$$

As discussed earlier, we first batch all quantization checks by Lemma D.5. We then prove the range of $\mathbf{r}$ with a lookup argument.

Segmented prefix sum. Given $f(X)$ encoding a vector $\mathbf{f}$ over $\mathbb{H}_{abc}$, we require three types of prefix sums: prove $\mathbf{g}$ and $\mathbf{h}$ such that

1. $\mathbf{g}[i,j,k] = \sum_{k'=0}^{k} \mathbf{f}[i,j,k']$ and $\mathbf{h}[i,j] = \sum_{k' \in [c]} \mathbf{f}[i,j,k']$;
2. $\mathbf{g}[i,j,k] = \sum_{j'=0}^{j} \mathbf{f}[i,j',k]$ and $\mathbf{h}[i,k] = \sum_{j' \in [b]} \mathbf{f}[i,j',k]$;
3. $\mathbf{g}[i,j,k] = \sum_{i'=0}^{i} \mathbf{f}[i',j,k]$ and $\mathbf{h}[j,k] = \sum_{i' \in [a]} \mathbf{f}[i',j,k]$.

We note that higher dimensional prefix sums can be reduced to one of the three types, e.g. the prefix sum along the second dimension of $\mathbf{V} \in \mathbb{F}^{a \times b \times c \times d}$ is reduced to proving the second type over $\mathbf{V}_{\{a \times b \times cd\}}$.

For brevity, we defer the concrete constructions to Appendix G.3.

Column-replication. Given $f(X)$ encoding $\mathbf{f} \in \mathbb{F}^n$ and $G(X)$ encoding $\mathbf{G} \in \mathbb{F}^{n \times m}$, the column-replication relation validates that

$$\forall i \in [n], j \in [m].\ \mathbf{G}[i][j] = \mathbf{f}[i]. \tag{E.8}$$

We defer the concrete construction to Appendix G.2.

E.2 Proof of Well-formedness

Following Section 5.3, we prove the well-formedness of the committed forest by proving the following properties.

Indicator consistency. The prover proves that **valid** agrees with the depth and tree indicators, and that invalid nodes are not marked as leaves:

$$\mathbf{valid}[i] = 1 \iff \mathbf{depth}[i] \neq 0, \tag{E.9}$$

$$\mathbf{valid}[i] = 1 \iff \mathbf{tree}[i] \neq 0, \tag{E.10}$$

$$\mathbf{valid}[i] = 0 \implies \mathbf{is_leaf}[i] = 0. \tag{E.11}$$

The equivalences are proved by lookups and zero-checks that bind the resulting indicator witnesses to **valid**.

Depth range. The prover commits $\mathbf{d}_{<h} \in \{0,1\}^{N_{\text{tot}}}$ and proves

$$\mathbf{depth} \| \mathbf{d}_{<h} \subseteq (0,1,\ldots,h)^\top \| (1,\ldots,1,0)^\top. \tag{E.12}$$

The public lookup table proves both $\mathbf{depth}[u] \in \{0,\ldots,h\}$ and $\mathbf{d}_{<h}[u] = 1 \iff \mathbf{depth}[u] < h$.

Boolean indicators. The prover uses zero-checks over $\mathbb{H}_{N_{\text{tot}}}$ to validate

$$\mathbf{root} \odot (\mathbf{root} - \mathbf{1}) = \mathbf{0}, \tag{E.13}$$

$$\mathbf{is_leaf} \odot (\mathbf{is_leaf} - \mathbf{1}) = \mathbf{0}, \tag{E.14}$$

$$\mathbf{valid} \odot (\mathbf{valid} - \mathbf{1}) = \mathbf{0}. \tag{E.15}$$

For readability, write

$$\mathbf{internal} := \mathbf{valid} \odot (\mathbf{1} - \mathbf{is_leaf})$$

for the valid non-leaf node indicator used below.

No ghost nodes. For every invalid node, all committed attributes are dummy. The prover proves by zero-checks over the appropriate domains that

$$(\mathbf{1} - \mathbf{valid}) \odot \mathbf{lc} = \mathbf{0}, \tag{E.16}$$
$$(\mathbf{1} - \mathbf{valid}) \odot \mathbf{rc} = \mathbf{0}, \tag{E.17}$$
$$(\mathbf{1} - \mathbf{valid}) \odot \mathbf{par} = \mathbf{0}, \tag{E.18}$$
$$(\mathbf{1} - \mathbf{valid}) \odot \mathbf{tree} = \mathbf{0}, \tag{E.19}$$
$$(\mathbf{1} - \mathbf{valid}) \odot \mathbf{root} = \mathbf{0}, \tag{E.20}$$
$$(\mathbf{1} - \mathbf{valid}) \odot \mathbf{is_leaf} = \mathbf{0}. \tag{E.21}$$

Let $\mathbf{Invalid} \in \mathbb{F}^{N_{\text{tot}} \times d}$ be generated by column-replicating d times from $\mathbf{1} - \mathbf{valid}$. The prover also checks

$$\mathbf{Invalid} \odot \mathbf{N}^{\leftarrow} = \mathbf{Invalid} \odot \mathbf{N}^{\rightarrow} = \mathbf{Invalid} \odot \mathbf{E} = \mathbf{0}. \tag{E.22}$$

Number of roots. To prove that the committed forest contains exactly TK roots, the prover uses sum-check over $\mathbb{H}_{N_{\text{tot}}}$ to prove

$$\sum_{i=0}^{N_{\text{tot}}-1} \mathbf{root}[i] = TK. \tag{E.23}$$

Depth of roots. The prover uses a zero-check over $\mathbb{H}_{N_{\text{tot}}}$ to validate

$$\mathbf{root} \odot (\mathbf{depth} - \mathbf{1}) = \mathbf{0}. \tag{E.24}$$

Uniqueness of roots. To prove that there is exactly one root for each tree identifier $1, \ldots, TK$, the prover commits $\mathbf{cnt} \in \{0,1\}^{TK+1}$, where $\mathbf{cnt}[t]$ counts the number of roots with tree identifier t. After these commitments are fixed and its booleanity is proved, the verifier samples α, and the prover proves

$$\sum_{t=1}^{TK} \alpha^t = \sum_{t \in [TK+1]} \alpha^t \cdot \mathbf{cnt}[t] = \sum_{u \in [N_{\text{tot}}]} \alpha^{\mathbf{tree}[u]} \cdot \mathbf{root}[u]. \tag{E.25}$$

The left-hand side is public. The prover commits the power table $\boldsymbol{\alpha} = (1, \alpha, \ldots, \alpha^{TK})$, proves

$$\boldsymbol{\alpha}[0] = 1, \tag{E.26}$$
$$\boldsymbol{\alpha}[i+1] = \alpha \cdot \boldsymbol{\alpha}[i], \ \forall 0 \le i < TK. \tag{E.27}$$

The prover then uses it to fetch $\alpha^{\mathbf{tree}[u]}$ for each node.

Parent-child consistency. The prover commits fetched-parent vectors $\mathbf{par_lc}$, $\mathbf{par_rc} \in \mathbb{F}^{N_{\text{tot}}}$ such that

$$\mathbf{par_lc}[u] = \mathbf{par}[\mathbf{lc}[u]], \qquad \mathbf{par_rc}[u] = \mathbf{par}[\mathbf{rc}[u]]. \tag{E.28}$$

Their correctness is proved by lookup:

$$\mathbf{lc} \| \mathbf{par_lc} \subseteq (0, 1, \ldots, N_{\text{tot}} - 1)^\top \| \mathbf{par}, \tag{E.29}$$

$$\mathbf{rc} \| \mathbf{par_rc} \subseteq (0, 1, \ldots, N_{\text{tot}} - 1)^\top \| \mathbf{par}. \tag{E.30}$$

The prover then checks, for internal nodes,

$$\mathbf{internal} \odot (\mathbf{par_lc} - (0, 1, \ldots, N_{\text{tot}} - 1)^\top) = \mathbf{0}, \tag{E.31}$$

$$\mathbf{internal} \odot (\mathbf{par_rc} - (0, 1, \ldots, N_{\text{tot}} - 1)^\top) = \mathbf{0}. \tag{E.32}$$

For the converse direction, the prover commits $\mathbf{lc_par}, \mathbf{rc_par} \in \mathbb{F}^{N_{\text{tot}}}$ such that

$$\mathbf{lc_par}[v] = \mathbf{lc}[\mathbf{par}[v]], \qquad \mathbf{rc_par}[v] = \mathbf{rc}[\mathbf{par}[v]], \tag{E.33}$$

and proves the corresponding lookup relations. It then checks

$$\begin{aligned} &\mathbf{valid} \odot (\mathbf{1} - \mathbf{root}) \\ \odot &(\mathbf{lc_par} - (0, 1, \ldots, N_{\text{tot}} - 1)^\top) \\ \odot &(\mathbf{rc_par} - (0, 1, \ldots, N_{\text{tot}} - 1)^\top) = \mathbf{0}. \end{aligned} \tag{E.34}$$

Finally, roots have dummy parent:

$$\mathbf{par} \odot \mathbf{root} = \mathbf{0}. \tag{E.35}$$

Tree index and depth consistency. The prover commits **tree_lc**, **tree_rc**, **depth_lc**, and **depth_rc** such that

$$\mathbf{tree_lc}[i] = \mathbf{tree}[\mathbf{lc}[i]], \tag{E.36}$$

$$\mathbf{tree_rc}[i] = \mathbf{tree}[\mathbf{rc}[i]], \tag{E.37}$$

$$\mathbf{depth_lc}[i] = \mathbf{depth}[\mathbf{lc}[i]], \tag{E.38}$$

$$\mathbf{depth_rc}[i] = \mathbf{depth}[\mathbf{rc}[i]]. \tag{E.39}$$

The prover proves their correctness by lookup:

$$\mathbf{lc} \| \mathbf{tree_lc} \subseteq (0, 1, \ldots, N_{\text{tot}} - 1)^\top \| \mathbf{tree}, \tag{E.40}$$

$$\mathbf{rc} \| \mathbf{tree_rc} \subseteq (0, 1, \ldots, N_{\text{tot}} - 1)^\top \| \mathbf{tree}, \tag{E.41}$$

$$\mathbf{lc} \| \mathbf{depth_lc} \subseteq (0, 1, \ldots, N_{\text{tot}} - 1)^\top \| \mathbf{depth}, \tag{E.42}$$

$$\mathbf{rc} \| \mathbf{depth_rc} \subseteq (0, 1, \ldots, N_{\text{tot}} - 1)^\top \| \mathbf{depth}. \tag{E.43}$$

The prover then uses zero-checks over $\mathbb{H}_{N_{\text{tot}}}$ to prove

$$\mathbf{internal} \odot (\mathbf{tree_lc} - \mathbf{tree}) = \mathbf{0}, \tag{E.44}$$

$$\mathbf{internal} \odot (\mathbf{tree_rc} - \mathbf{tree}) = \mathbf{0}, \tag{E.45}$$

$$\mathbf{internal} \odot (\mathbf{depth_lc} - \mathbf{depth} - \mathbf{1}) = \mathbf{0}, \tag{E.46}$$

$$\mathbf{internal} \odot (\mathbf{depth_rc} - \mathbf{depth} - \mathbf{1}) = \mathbf{0}. \tag{E.47}$$

Since all depths are range-checked in $\{0, \ldots, h\}$, these constraints force every node at depth h to be a leaf. It also proves that the two children of every internal node are distinct by range-checking the nonzero difference $\mathbf{lc} - \mathbf{rc}$ on internal nodes.

Consistency of leaf nodes. To prove that leaves have no child, the prover uses zero-check over $\mathbb{H}_{N_{\text{tot}}}$ to validate

$$\mathbf{is_leaf} \odot \mathbf{lc} = \mathbf{0}, \tag{E.48}$$

$$\mathbf{is_leaf} \odot \mathbf{rc} = \mathbf{0}. \tag{E.49}$$

Hyper-box of roots. To prove that the hyper-box of any root is full, the prover uses zero-check over $\mathbb{H}_{N_{\text{tot}}d}$ to validate

$$\mathbf{Root} \odot \mathbf{N}^{\leftarrow} = \mathbf{0}, \tag{E.50}$$

$$\mathbf{Root} \odot \left(\mathbf{N}^{\rightarrow} - B \cdot \mathbf{1}\right) = \mathbf{0}. \tag{E.51}$$

Nonempty valid boxes. The prover proves by range lookup that, for every valid node and every feature,

$$\mathbf{N}^{\rightarrow}[u, j] - \mathbf{N}^{\leftarrow}[u, j] > 0. \tag{E.52}$$

Hyper-box consistency of children. To prove that the child hyper-boxes are consistent with the parent and the selected split, the prover first commits $\mathbf{N}_{\text{lc}}^{\leftarrow}, \mathbf{N}_{\text{lc}}^{\rightarrow}, \mathbf{N}_{\text{rc}}^{\leftarrow}, \mathbf{N}_{\text{rc}}^{\rightarrow} \in \mathbb{F}^{N_{\text{tot}} \times d}$ and proves that

$$\mathbf{lc} \| \mathbf{N}_{\text{lc}}^{\leftarrow} \| \mathbf{N}_{\text{lc}}^{\rightarrow} \subseteq (0, 1, \ldots, N_{\text{tot}} - 1)^{\top} \| \mathbf{N}^{\leftarrow} \| \mathbf{N}^{\rightarrow}, \tag{E.53}$$

$$\mathbf{rc} \| \mathbf{N}_{\text{rc}}^{\leftarrow} \| \mathbf{N}_{\text{rc}}^{\rightarrow} \subseteq (0, 1, \ldots, N_{\text{tot}} - 1)^{\top} \| \mathbf{N}^{\leftarrow} \| \mathbf{N}^{\rightarrow}. \tag{E.54}$$

Let $\mathbf{Internal} \in \mathbb{F}^{N_{\text{tot}} \times d}$ be generated by column-replication d times from **internal**. The prover then uses zero-check over $\mathbb{H}_{N_{\text{tot}}d}$ to validate

$$\mathbf{Internal} \odot \left(\mathbf{N}^{\leftarrow} - \mathbf{N}_{\text{lc}}^{\leftarrow}\right) = \mathbf{0}, \tag{E.55}$$

$$\mathbf{Internal} \odot \left(\mathbf{N}^{\rightarrow} - \mathbf{N}_{\text{rc}}^{\rightarrow}\right) = \mathbf{0}, \tag{E.56}$$

$$\mathbf{Internal} \odot \mathbf{E} \odot \left(\mathbf{N}_{\text{lc}}^{\rightarrow} - \mathbf{N}_{\text{rc}}^{\leftarrow}\right) = \mathbf{0}, \tag{E.57}$$

$$\mathbf{Internal} \odot (\mathbf{1} - \mathbf{E}) \odot \left(\mathbf{N}^{\leftarrow} - \mathbf{N}_{\text{rc}}^{\leftarrow}\right) = \mathbf{0}, \tag{E.58}$$

$$\mathbf{Internal} \odot (\mathbf{1} - \mathbf{E}) \odot \left(\mathbf{N}^{\rightarrow} - \mathbf{N}_{\text{lc}}^{\rightarrow}\right) = \mathbf{0}. \tag{E.59}$$

Note that the above equations hold for dummy node 0, since all its attributes are zero.

Feature-selection matrix. To prove that $\mathbf{E}$ is boolean and one-hot on internal nodes and zero on leaves, the prover first uses zero-check over $\mathbb{H}_{N_{\text{tot}}d}$ to validate

$$\mathbf{E} \odot (\mathbf{E} - \mathbf{1}) = \mathbf{0}. \tag{E.60}$$

It then invokes the prefix-sum primitive along the feature dimension to obtain $\mathbf{E}_{\text{sum}} \in \mathbb{F}^{N_{\text{tot}}}$, where

$$\mathbf{E}_{\text{sum}}[i] = \sum_{j \in [d]} \mathbf{E}[i, j]. \tag{E.61}$$

Finally, the prover uses zero-check over $\mathbb{H}_{N_{\text{tot}}}$ to validate

$$\mathbf{E}_{\text{sum}} - \mathbf{internal} = \mathbf{0}. \tag{E.62}$$

Scores of internal nodes. To prove that internal nodes carry no leaf score, the prover uses zero-check over $\mathbb{H}_{N_{\text{tot}}}$ to validate

$$\mathbf{score} \odot (\mathbf{1} - \mathbf{valid} \odot \mathbf{is_leaf}) = \mathbf{0}. \tag{E.63}$$

E.3 Proof of Inference

Following Equation (5.1), the prover proves batched inference for public inputs $\mathbf{I} \in [B]^{b \times d}$ and claimed one-hot labels $\mathbf{Y} \in \{0,1\}^{K \times b}$ by proving the following properties.

Leaf lookup. The prover commits $\mathbf{U}^{\leftarrow}, \mathbf{U}^{\rightarrow} \in \mathbb{F}^{TKb \times d}$ and $\mathbf{sc} \in \mathbb{F}^{TKb}$. Let $\mathbf{ind_tree} \in [TK+1]^{TKb}$ be the public vector

$$\mathbf{ind_tree}_{\{T \times K \times b\}}[t,k,i] = tK + k + 1, \tag{E.64}$$

consistent with the convention that tree identifier 0 is dummy, and that the logical pair (t,k) denotes the real tree whose identifier is $tK+k+1$. The prover proves the lookup relation

$$\mathbf{1} \| \mathbf{ind_tree} \| \mathbf{U}^{\leftarrow} \| \mathbf{U}^{\rightarrow} \| \mathbf{sc} \subseteq \mathbf{is_leaf} \| \mathbf{tree} \| \mathbf{N}^{\leftarrow} \| \mathbf{N}^{\rightarrow} \| \mathbf{score}. \tag{E.65}$$

The leading all-one column forces each selected node to be a leaf, while the tree-index column forces the selected leaf to belong to the intended tree. Since the bounds and score are fetched in the same lookup row, this also binds each row of $\mathbf{U}^{\leftarrow}, \mathbf{U}^{\rightarrow}$ and $\mathbf{sc}$ to the same committed leaf.

Interval containment. Let $\mathbf{I}' \in \mathbb{F}^{TKb \times d}$ be obtained by row-replicating $\mathbf{I}$ TK times, i.e. for each t,k,i,j,

$$\mathbf{I}'_{\{T \times K \times b \times d\}}[t,k,i,j] = \mathbf{I}[i,j]. \tag{E.66}$$

By Lemma D.3, we have $I'(X) = I(X^{TK})$, so the prover does not need to commit $\mathbf{I}'$.

To prove that every selected leaf contains the corresponding input sample, the prover forms

$$\boldsymbol{\Delta}^{\leftarrow} = \mathbf{I}' - \mathbf{U}^{\leftarrow}, \tag{E.67}$$

$$\boldsymbol{\Delta}^{\rightarrow} = \mathbf{U}^{\rightarrow} - \mathbf{I}' - \mathbf{1}, \tag{E.68}$$

and proves by range lookup over $\{0,\dots,B\}$ that all entries of $\boldsymbol{\Delta}^{\leftarrow}$ and $\boldsymbol{\Delta}^{\rightarrow}$ are in range. Equivalently, for every t,k,i,j,

$$\mathbf{U}^{\leftarrow}_{\{T \times K \times b \times d\}}[t,k,i,j] \le \mathbf{I}[i,j] < \mathbf{U}^{\rightarrow}_{\{T \times K \times b \times d\}}[t,k,i,j]. \tag{E.69}$$

Together with forest well-formedness, this proves that each entry of $\mathbf{sc}_{\{T \times K \times b\}}$ is the score assigned to that sample by the corresponding tree.

Score aggregation. The prover commits $\mathbf{S} \in \mathbb{F}^{K\times b}$, where $\mathbf{S}[k,i]$ is the accumulated inference score of sample i for class k. It invokes the prefix-sum primitive along the round dimension of $\mathbf{sc}_{\{T\times K\times b\}}$ to prove

$$\mathbf{S}[k,i] = \sum_{t=0}^{T-1} \mathbf{sc}_{\{T\times K\times b\}}[t,k,i], \qquad \forall k \in [K], i \in [b]. \tag{E.70}$$

Maximum selection. To prove that $\mathbf{Y}$ contains the predicted class labels, the verifier first checks directly that $\mathbf{Y}$ is one-hot. The prover commits a masked score matrix $\widehat{\mathbf{S}} \in \mathbb{F}^{K\times b}$ and proves by zero-check over $\mathbb{H}_{Kb}$ that

$$\widehat{\mathbf{S}} - \mathbf{Y} \odot \mathbf{S} = \mathbf{0}. \tag{E.71}$$

The prover then invokes the prefix-sum primitive along the class dimension to obtain $\mathbf{s} \in \mathbb{F}^b$ and proves

$$\mathbf{s}[i] = \sum_{k=0}^{K-1} \widehat{\mathbf{S}}[k,i], \qquad \forall i \in [b]. \tag{E.72}$$

Row-replicating $\mathbf{s}$ K times to obtain $\mathbf{S}' \in \mathbb{F}^{K\times b}$, the prover finally proves

$$0 \le \mathbf{S}' - \mathbf{S} < \mathcal{B}, \tag{E.73}$$

which certifies that $\mathbf{S}'$ selects the maximum score in each column of $\mathbf{S}$, and thus certifies the correctness of $\mathbf{Y}$.

E.4 Proof of Training

The prover commits the training dataset $\mathbf{D} \in [B]^{M\times d}$, the labels $\mathbf{Y} \in \{0,1\}^{K\times M}$, the pseudo-residuals $\mathbf{G} \in \mathbb{F}^{T\times K\times M}$, and $\mathbf{Ind} \in [N_{\text{tot}}]^{T\times K\times M}$ which indicates where a sample falls in each tree. Range and zero-checks prove

$$\mathbf{D}[r,j] \in [B], \tag{E.74}$$

$$\mathbf{Y}[k,r](\mathbf{Y}[k,r]-1) = 0, \tag{E.75}$$

$$\sum_{k\in[K]} \mathbf{Y}[k,r] = 1, \tag{E.76}$$

and that every entry of $\mathbf{S}$ and $\mathbf{G}$ lies in the signed code space $[-\mathcal{B}/2, \mathcal{B}/2)$. The proof then certifies each trained tree in the following order.

Correctness of $\mathbf{G}$. Recall that we require

$$\begin{aligned}
\mathbf{G}[t,k,r] &= \begin{cases} 2^q\mathbf{Y}[k,r] - \lfloor 2^q/K \rfloor & \text{if } t=0, \\ 2^q\mathbf{Y}[k,r] - \mathbf{S}'[t-1,k,r] & \text{if } t>0. \end{cases}, \\
\mathbf{S}'[t,k,r] &= \left\lfloor \frac{\exp(\mathbf{S}[t,k,r] - \mathbf{S}_{\max}[t,k,r]) \cdot 2^q}{\sum_{k'=0}^{K-1} \exp(\mathbf{S}[t,k',r] - \mathbf{S}_{\max}[t,k',r])} \right\rfloor, \\
\mathbf{S}_{\max}[t,k,r] &= \max_{k'\in[K]} \mathbf{S}[t,k',r].
\end{aligned} \tag{E.77}$$

The correctness of $\mathbf{S}_{\max}$ is proved in a same manner as in inference. Quantized exponentials are proved by looking up the public table defined in Section 2.3; its commitment is input-independent preprocessing. Quantized divisions are proved by committing and range-checking the remainder.

Dataset leaf assignment. The prover commits the bound tensors

$$\mathbf{U}^{\leftarrow}, \mathbf{U}^{\rightarrow} \in \mathbb{F}^{TKM \times d}$$

and the score vector $\mathbf{sc} \in \mathbb{F}^{TKM}$. Let

$$\mathbf{ind_tree}_{\{T\times K\times M\}}[t,k,r] = tK + k + 1, \tag{E.78}$$

which is obtained by column-replicating $(1, 2, \ldots, TK)^\top$ M times. Thus the logical pair (t, k) denotes the real tree whose identifier is $tK + k + 1$. The prover proves

$$\mathbf{Ind}_{\{TKM\}} \| \mathbf{1} \| \mathbf{ind_tree} \| \mathbf{U}^{\leftarrow} \| \mathbf{U}^{\rightarrow} \| \mathbf{sc} \subseteq \tag{E.79}$$

$$(0, 1, \ldots, N_{\text{tot}} - 1)^\top \| \mathbf{is_leaf} \| \mathbf{tree} \| \mathbf{N}^{\leftarrow} \| \mathbf{N}^{\rightarrow} \| \mathbf{score}, \tag{E.80}$$

where the committed vector $\mathbf{Ind} \in [N_{\text{tot}}]^{T\times K\times M}$ indicates the leaf to which each training sample falls in each tree, and $\mathbf{sc}$ indicates the score each sample receives from each tree. It then row-replicates $\mathbf{D}$ TK times and proves by range lookup that, for all t, k, r, j,

$$\mathbf{U}^{\leftarrow}_{\{T\times K\times M\times d\}}[t,k,r,j] \le \mathbf{D}[r,j] < \mathbf{U}^{\rightarrow}_{\{T\times K\times M\times d\}}[t,k,r,j]. \tag{E.81}$$

This proves that $\mathbf{Ind}[t, k, r]$ is the leaf reached by sample r in tree (t, k).

Correctness of $\mathbf{S}$. This is proved by proving prefix-sum, i.e.

$$\mathbf{S}[t,k,r] = \sum_{t'=0}^{t} \mathbf{sc}_{\{T\times K\times M\}}[t',k,r], \quad \forall t \in [T], k \in [K], r \in [M]. \tag{E.82}$$

Histogram. The prover commits leaf histograms $\mathbf{C}_{\text{leaf}}, \mathbf{V}_{\text{leaf}} \in \mathbb{F}^{N_{\text{tot}}\times d\times B}$. It also commits an address tensor $\mathbf{A} \in \mathbb{F}^{T\times K\times M\times d}$ and proves its correctness by zero-check:

$$\mathbf{A}[t,k,r,j] - \mathbf{Ind}[t,k,r] \cdot dB - j \cdot B - \mathbf{D}[r,j] = 0. \tag{E.83}$$

Thus each row of $\mathbf{A}$ is one routed sample-feature update into the node-feature-bin address space of size $N_{\text{tot}}dB$.

After $\mathbf{A}, \mathbf{C}_{\text{leaf}}, \mathbf{V}_{\text{leaf}}$ are fixed, the verifier samples α. The prover commits $\mathbf{P} = (1, \alpha, \ldots, \alpha^{N_{\text{tot}}dB-1})$ and proves

$$P(\omega^0_{N_{\text{tot}}dB}) = 1, \tag{E.84}$$

$$\left(1 - \lambda^{(0)}_{N_{\text{tot}}dB}(X)\right)\left(P(X) - \alpha P(\omega^{-1}_{N_{\text{tot}}dB}X)\right) = q(X) \cdot \nu_{N_{\text{tot}}dB}(X). \tag{E.85}$$

Thus $\mathbf{P}[i] = \alpha^i$ for every address i.

The prover then creates $\mathbf{A}'[t,k,r,j] = \alpha^{\mathbf{A}[t,k,r,j]}$ and proves its correctness by lookup:

$$\mathbf{A} \| \mathbf{A}' \subseteq (0, 1, \ldots, N_{\text{tot}} dB - 1)^\top \| \mathbf{P}. \tag{E.86}$$

The prover then proves the correctness of the leaf histograms by batching the following sum-check identities over all feature-bin columns:

$$\begin{aligned} &\sum_{u,j,b} \mathbf{C}_{\text{leaf}}[u,j,b] \cdot \mathbf{P}_{\{N_{\text{tot}} \times d \times B\}}[u,j,b] \\ &= \sum_{t,k,r,j} \mathbf{A}'[t,k,r,j], \end{aligned} \tag{E.87}$$

$$\begin{aligned} &\sum_{u,j,b} \mathbf{V}_{\text{leaf}}[u,j,b] \cdot \mathbf{P}_{\{N_{\text{tot}} \times d \times B\}}[u,j,b] \\ &= \sum_{t,k,r,j} \mathbf{A}'[t,k,r,j] \cdot \mathbf{G}[t,k,r]. \end{aligned} \tag{E.88}$$

The prover then commits full histograms $\mathbf{C}, \mathbf{V} \in \mathbb{F}^{N_{\text{tot}} \times d \times B}$. It commits routed child histograms $\mathbf{C}_{\text{lc}}, \mathbf{C}_{\text{rc}}, \mathbf{V}_{\text{lc}}, \mathbf{V}_{\text{rc}} \in \mathbb{F}^{N_{\text{tot}} \times d \times B}$, and proves their correctness by lookup:

$$\mathbf{lc} \| \mathbf{C}_{\text{lc}} \| \mathbf{V}_{\text{lc}} \subseteq (0, 1, \ldots, N_{\text{tot}} - 1)^\top \| \mathbf{C} \| \mathbf{V}, \tag{E.89}$$

$$\mathbf{rc} \| \mathbf{C}_{\text{rc}} \| \mathbf{V}_{\text{rc}} \subseteq (0, 1, \ldots, N_{\text{tot}} - 1)^\top \| \mathbf{C} \| \mathbf{V}. \tag{E.90}$$

Let $\mathbf{Leaf} \in \mathbb{F}^{N_{\text{tot}} \times d \times B}$ be generated by column-replication dB times from $\mathbf{is_leaf}$. The prover then uses zero-check over $\mathbb{H}_{N_{\text{tot}} dB}$ to validate

$$\mathbf{C} - \mathbf{Leaf} \odot \mathbf{C}_{\text{leaf}} - (\mathbf{1} - \mathbf{Leaf}) \odot \left(\mathbf{C}_{\text{lc}} + \mathbf{C}_{\text{rc}}\right) = 0, \tag{E.91}$$

$$\mathbf{V} - \mathbf{Leaf} \odot \mathbf{V}_{\text{leaf}} - (\mathbf{1} - \mathbf{Leaf}) \odot \left(\mathbf{V}_{\text{lc}} + \mathbf{V}_{\text{rc}}\right) = 0. \tag{E.92}$$

Leaf values. The prover now proves that the score assigned to each leaf is consistent with the learning-rate-scaled mean pseudo-residual of the samples routed to that leaf. The prover commits two vectors $\mathbf{n}_{\text{leaf}}, \mathbf{v}_{\text{leaf}} \in \mathbb{F}^{N_{\text{tot}}}$ and proves, by proving prefix-sum, that for every node u,

$$\mathbf{n}_{\text{leaf}}[u] = \sum_{i \in [dB]} \mathbf{C}_{\text{leaf},\{N_{\text{tot}} \times dB\}}[u,i], \tag{E.93}$$

$$\mathbf{v}_{\text{leaf}}[u] = \sum_{i \in [dB]} \mathbf{V}_{\text{leaf},\{N_{\text{tot}} \times dB\}}[u,i]. \tag{E.94}$$

Because each routed sample contributes exactly one update to every feature coordinate, $\mathbf{n}_{\text{leaf}}[u]$ is d times the number of training samples assigned to leaf u, and $\mathbf{v}_{\text{leaf}}[u]$ is d times the corresponding sum of pseudo-residuals. This common

factor cancels in the leaf-value ratio below. Using the public fixed-point learning rate μ_q, the leaf score should be

$$\mathbf{score}[u] = \left\lfloor \frac{\mu_q \cdot \mathbf{v}_{\text{leaf}}[u]}{2^q \cdot \mathbf{n}_{\text{leaf}}[u]} \right\rfloor$$

on valid leaves. To avoid a division in the arithmetic relation, the prover commits a remainder vector $\mathbf{r}_{\text{leaf}} \in \mathbb{F}^{N_{\text{tot}}}$ and proves by zero-check over $\mathbb{H}_{N_{\text{tot}}}$ that

$$\mathbf{is_leaf} \odot (\mu_q \cdot \mathbf{v}_{\text{leaf}} - \mathbf{score} \odot \mathbf{n}_{\text{leaf}} \cdot 2^q - \mathbf{r}_{\text{leaf}}) = \mathbf{0}. \tag{E.95}$$

It also proves

$$(\mathbf{1} - \mathbf{is_leaf}) \odot \mathbf{r}_{\text{leaf}} = \mathbf{0} \tag{E.96}$$

and applies range lookups to the values masked by **is_leaf** to prove

$$0 \le \mathbf{r}_{\text{leaf}}[u] < \mathcal{B}, \tag{E.97}$$

$$0 \le 2^q \mathbf{n}_{\text{leaf}}[u] - \mathbf{r}_{\text{leaf}}[u] - 1 < \mathcal{B}. \tag{E.98}$$

The strict remainder condition also enforces $\mathbf{n}_{\text{leaf}}[u] > 0$ for every valid leaf.

Optimal split. The prover invokes the prefix-sum primitive along the bin dimension to obtain $\mathbf{C}_<, \mathbf{V}_<$ and $\mathbf{C}_\ge, \mathbf{V}_\ge$. The prover commits non-zero indicators $\boldsymbol{\chi}_<, \boldsymbol{\chi}_\ge \in \{0,1\}^{N_{\text{tot}} \times d \times B}$. The prover proves their correctness by proving

$$\mathbf{C}_< \| \boldsymbol{\chi}_< \subseteq \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ \vdots & \vdots \\ M & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{C}_\ge \| \boldsymbol{\chi}_\ge \subseteq \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ \vdots & \vdots \\ M & 1 \end{pmatrix} \tag{E.99}$$

The prover commits a valid-split indicator $\boldsymbol{\chi} \in \{0,1\}^{N_{\text{tot}} \times d \times B}$ and proves

$$\boldsymbol{\chi} - \boldsymbol{\chi}_< \odot \boldsymbol{\chi}_\ge = \mathbf{0}. \tag{E.100}$$

Thus, $\boldsymbol{\chi}[u, j, b] = 1$ exactly when both children are nonempty if we split node u by (j, b). The prover also commits the left and right quotient tensors $\boldsymbol{\Gamma}_<, \boldsymbol{\Gamma}_\ge \in \mathbb{F}^{N_{\text{tot}} \times d \times B}$, their remainder tensors $\mathbf{R}_<, \mathbf{R}_\ge \in \mathbb{F}^{N_{\text{tot}} \times d \times B}$, and the split-score tensor $\boldsymbol{\Gamma} \in \mathbb{F}^{N_{\text{tot}} \times d \times B}$. With public fixed-point scale 2^q, the prover proves by zero-check that

$$\boldsymbol{\chi} \odot \left((\mathbf{V}_<)^2 - \boldsymbol{\Gamma}_< \odot \mathbf{C}_< \cdot 2^q - \mathbf{R}_< \right) = \mathbf{0}, \tag{E.101}$$

$$\boldsymbol{\chi} \odot \left((\mathbf{V}_\ge)^2 - \boldsymbol{\Gamma}_\ge \odot \mathbf{C}_\ge \cdot 2^q - \mathbf{R}_\ge \right) = \mathbf{0}, \tag{E.102}$$

$$\boldsymbol{\chi} \odot \left(\boldsymbol{\Gamma} - \boldsymbol{\Gamma}_< - \boldsymbol{\Gamma}_\ge \right) = \mathbf{0}, \tag{E.103}$$

$$(\mathbf{1} - \boldsymbol{\chi}) \odot \boldsymbol{\Gamma} = (\mathbf{1} - \boldsymbol{\chi}) \odot \boldsymbol{\Gamma}_< = (\mathbf{1} - \boldsymbol{\chi}) \odot \boldsymbol{\Gamma}_\ge = \mathbf{0}. \tag{E.104}$$

It proves the Euclidean-remainder conditions

$$0 \le \mathbf{R}_{<}[u,j,b] < 2^q \mathbf{C}_{<}[u,j,b], \tag{E.105}$$
$$0 \le \mathbf{R}_{\ge}[u,j,b] < 2^q \mathbf{C}_{\ge}[u,j,b], \tag{E.106}$$

for every valid candidate by range/lookup checks. Concretely, the prover also checks that the strict slacks

$$2^q \mathbf{C}_{<}[u,j,b] - \mathbf{R}_{<}[u,j,b] - 1, \qquad 2^q \mathbf{C}_{\ge}[u,j,b] - \mathbf{R}_{\ge}[u,j,b] - 1$$

lie in the public range table. Thus $\boldsymbol{\Gamma}_{<}$ and $\boldsymbol{\Gamma}_{\ge}$ are the quantized division results

$$\boldsymbol{\Gamma}_{<} = \left\lfloor \frac{(\mathbf{V}_{<})^2}{2^q \mathbf{C}_{<}} \right\rfloor \text{ and } \boldsymbol{\Gamma}_{\ge} = \left\lfloor \frac{(\mathbf{V}_{\ge})^2}{2^q \mathbf{C}_{\ge}} \right\rfloor,$$

and define $\boldsymbol{\Gamma}$ as their sum:

$$\boldsymbol{\Gamma} = \boldsymbol{\Gamma}_{<} + \boldsymbol{\Gamma}_{\ge}. \tag{E.107}$$

Define and prove $\mathbf{C}', \mathbf{V}' \in \mathbb{F}^{N_{\text{tot}}}$ by

$$\mathbf{C}'[u] = d^{-1} \sum_{j \in [d]} \sum_{b \in [B]} \mathbf{C}[u,j,b], \tag{E.108}$$
$$\mathbf{V}'[u] = d^{-1} \sum_{j \in [d]} \sum_{b \in [B]} \mathbf{V}[u,j,b]. \tag{E.109}$$

The prover commits $\boldsymbol{\Gamma}_{\text{leaf}}, \mathbf{R}_{\text{leaf}} \in \mathbb{F}^{N_{\text{tot}}}$ and proves, for each valid node,

$$\mathbf{V}'[u]^2 = 2^q \mathbf{C}'[u] \boldsymbol{\Gamma}_{\text{leaf}}[u] + \mathbf{R}_{\text{leaf}}[u], \tag{E.110}$$
$$0 \le \mathbf{R}_{\text{leaf}}[u] < 2^q \mathbf{C}'[u]. \tag{E.111}$$

Thus $\boldsymbol{\Gamma}_{\text{leaf}}[u] = \lfloor \mathbf{V}'[u]^2 / (2^q \mathbf{C}'[u]) \rfloor$. The prover also commits an eligibility mask $\boldsymbol{\zeta} \in \{0,1\}^{N_{\text{tot}} \times d \times B}$, which indicates if a split is valid and has a gain above the threshold. The prover first proves its Booleanity and

$$\boldsymbol{\zeta} \odot (\mathbf{1} - \boldsymbol{\chi}) = \mathbf{0}. \tag{E.112}$$

It then uses masked range lookups to prove

$$0 \le \boldsymbol{\Gamma}[u,j,b] - \boldsymbol{\Gamma}_{\text{leaf}}[u] - \tau_q < \mathcal{B} \qquad \text{if } \boldsymbol{\zeta}[u,j,b] = 1, \tag{E.113}$$
$$0 \le \boldsymbol{\Gamma}_{\text{leaf}}[u] + \tau_q - \boldsymbol{\Gamma}[u,j,b] - 1 < \mathcal{B} \qquad \text{if } (\boldsymbol{\chi} - \boldsymbol{\zeta})[u,j,b] = 1. \tag{E.114}$$

Since $\boldsymbol{\chi}$ and $\boldsymbol{\zeta}$ are Boolean and Equation (E.112) gives $\boldsymbol{\zeta} \le \boldsymbol{\chi}$ entry-wise, the mask $\boldsymbol{\chi} - \boldsymbol{\zeta}$ is also Boolean. Hence Equations (E.113) and (E.114) prove the exact equivalence

$$\boldsymbol{\zeta}[u,j,b] = 1 \quad \Longleftrightarrow \quad \boldsymbol{\chi}[u,j,b] = 1 \ \wedge \ \boldsymbol{\Gamma}[u,j,b] \ge \boldsymbol{\Gamma}_{\text{leaf}}[u] + \tau_q.$$

The prover commits the score of the selected split $\boldsymbol{\Gamma}^* \in \mathbb{F}^{N_{\text{tot}}}$ and a selector $\mathbf{T} \in \{0,1\}^{N_{\text{tot}} \times d \times B}$. It proves

$$\mathbf{T} \odot (\mathbf{T} - \mathbf{1}) = \mathbf{0}, \tag{E.115}$$

$$\sum_{j \in [d]} \sum_{b \in [B]} \mathbf{T}[u,j,b] = \mathbf{internal}[u]. \tag{E.116}$$

$$\forall u.\ \boldsymbol{\Gamma}^*[u] = \sum_{j \in [d]} \sum_{b \in [B]} \mathbf{T}[u,j,b] \boldsymbol{\Gamma}[u,j,b]. \tag{E.117}$$

The selected candidate is required to be eligible:

$$\forall u.\ \sum_{j \in [d]} \sum_{b \in [B]} \mathbf{T}[u,j,b] \boldsymbol{\zeta}[u,j,b] = \mathbf{internal}[u]. \tag{E.118}$$

With $\mathbf{d}_{<h}$ from Equation (E.12), the prover proves the stopping condition

$$\mathbf{is_leaf}[u] \mathbf{d}_{<h}[u] \sum_{j \in [d]} \sum_{b \in [B]} \boldsymbol{\zeta}[u,j,b] = 0, \qquad \forall u \in [N_{\text{tot}}]. \tag{E.119}$$

This proves that a leaf either has reached the maximum depth, has no non-empty split, or has no splits with sufficient gains. For every internal node and valid candidate (j,b), the prover masks the following difference with the Boolean mask $\mathbf{internal}[u]\boldsymbol{\chi}[u,j,b]$ and proves by range lookup that

$$\begin{aligned} 0 \le \big(\mathbf{internal}[u] \cdot \boldsymbol{\chi}[u,j,b]\big) \cdot \\ \Big(dB\big(\boldsymbol{\Gamma}^*[u] - \boldsymbol{\Gamma}[u,j,b]\big) + jB + b \\ - \sum_{j' \in [d]} \sum_{b' \in [B]} \mathbf{T}[u,j',b'](j'B + b') \Big) < \mathcal{B}. \end{aligned} \tag{E.120}$$

Because candidate indices differ by less than dB, this proves score optimality with smallest-index tie-breaking. The no-wrap-around bound in Equation (2.11), together with the Booleanity of $\boldsymbol{\zeta}$, also makes a zero sum in Equation (E.119) equivalent to every entry in that row being zero. Hence a node below depth h is split exactly when an eligible candidate exists, and the selected candidate maximizes the fixed-point encoding of Φ.

Applying the selected split. The prover binds $\mathbf{T}$ with the one-hot forest feature mask $\mathbf{E}$ by proving

$$\mathbf{E}[u,j] = \sum_{b \in [B]} \mathbf{T}[u,j,b]. \tag{E.121}$$

It then links the selected threshold directly to the routed child bounds already committed in the well-formedness proof. For every $u \in [N_{\text{tot}}]$ and $j \in [d]$, it

validates

$$\mathbf{internal}[u]\mathbf{E}[u,j]\left(\mathbf{N}_{\mathrm{lc}}^{\rightarrow}[u,j]-\sum_{b\in[B]} b\,\mathbf{T}[u,j,b]\right)=0, \tag{E.122}$$

$$\mathbf{internal}[u]\mathbf{E}[u,j]\left(\mathbf{N}_{\mathrm{rc}}^{\leftarrow}[u,j]-\sum_{b\in[B]} b\,\mathbf{T}[u,j,b]\right)=0. \tag{E.123}$$

Together with the maximum proof above and the hyper-box consistency checks, this proves that the optimal split selected from the histograms is exactly the split applied in the committed forest.

F Tree Ensemble Extensions

This appendix records how the proof skeleton adapts to nearby tree-ensemble trainers. In both extensions, the forest encoding, leaf-routing lookups, well-formedness checks, histogram construction, and the two batching techniques remain unchanged. What changes is the plaintext trainer: the training target, the split objective, the leaf-value formula, and the round update.

F.1 Extending to XGBoost

Statement. The public statement fixes an XGBoost-style [8] quantized trainer, including the loss function, learning rate η, regularization parameters λ and γ, tree-shape policy, quantization parameters, and all dimensions. The prover commits to the dataset, labels, forest, leaf values, and any auxiliary information. The verifier should be convinced that the committed forest is obtained by running this public XGBoost-style trainer on the committed data.

Plaintext trainer. XGBoost also trains an additive model over rounds, but the tree for each class is fitted using a second-order approximation of the loss [8]. As in Equation (2.10), write $\mathbf{S}[t-1,k,r]$ for the score of sample r and class k before the tree of round t is added, with $\mathbf{S}[-1,k,r]=0$. The public trainer fixes the loss and therefore fixes the sample-wise gradient and Hessian values

$$g_{t,k,r}=\frac{\partial\mathcal{L}^{(t-1)}}{\partial\mathbf{S}[t-1,k,r]},\qquad h_{t,k,r}=\frac{\partial^2\mathcal{L}^{(t-1)}}{\partial\mathbf{S}[t-1,k,r]^2}.$$

For multiclass classification, we use a class-wise diagonal Hessian. Thus each class-k tree in round t is trained from $(g_{t,k,r},h_{t,k,r})_{r\in[M]}$.

Fix a node u in the class-k tree of round t, and let I_u be the samples reaching it. For a candidate split specified by feature j and threshold bin b, define

$$L_{u,j,b}:=\{\,r\in I_u:\mathbf{D}[r,j]<b\,\}, \tag{F.1}$$

$$R_{u,j,b}:=\{\,r\in I_u:\mathbf{D}[r,j]\geq b\,\}. \tag{F.2}$$

For any sample set $A \subseteq [M]$, write

$$G_A := \sum_{r \in A} g_{t,k,r}, \qquad H_A := \sum_{r \in A} h_{t,k,r}.$$

Define $G_u = G_{I_u}$, $H_u = H_{I_u}$, $G_L = G_{L_{u,j,b}}$, $H_L = H_{L_{u,j,b}}$, $G_R = G_{R_{u,j,b}}$, and $H_R = H_{R_{u,j,b}}$. The proof uses the following score, which is twice the usual XGBoost gain [8] and therefore gives the same ordering and pruning decision:

$$\Phi_{\mathtt{xgb}}(u, j, b) = \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G_u^2}{H_u + \lambda} - 2\gamma. \tag{F.3}$$

The trainer selects the valid candidate with largest positive score; if no such candidate exists, the node becomes a leaf. Once the tree structure is fixed, the leaf value of a leaf ℓ with sample set I_ℓ is

$$w_\ell = -\frac{G_{I_\ell}}{H_{I_\ell} + \lambda}. \tag{F.4}$$

The score update is

$$\mathbf{S}[t, k, r] = \mathbf{S}[t-1, k, r] + \eta \cdot w_{\ell^{(t,k)}(r)},$$

where $\ell^{(t,k)}(r)$ is the leaf reached by sample r in the tree trained at round t for class k.

Proof sketch. For multi-class logistic loss, the gradient and Hessian in round t are computed from the current prediction scores, which depend on all trees trained in the previous rounds. More precisely, let $\mathbf{S}[t-1, k, r]$ denote the score of sample r for class k before round t starts. It is the accumulated output of the class-k trees from all earlier rounds:

$$\mathbf{S}[t-1, k, r] = \sum_{\tau < t} \eta \cdot w_{\ell^{(\tau,k)}(r)},$$

where $\ell^{(\tau,k)}(r)$ is the leaf reached by sample r in the class-k tree trained at round τ, and $w_{\ell^{(\tau,k)}(r)}$ is the corresponding leaf value. We set $\mathbf{S}[-1, k, r] = 0$ if the rounds are indexed from 0.

The plaintext trainer first applies softmax to these accumulated scores:

$$\mathbf{P}[t, k, r] = \frac{\exp(\mathbf{S}[t-1, k, r])}{\sum_{c \in [K]} \exp(\mathbf{S}[t-1, c, r])}.$$

Under the standard diagonal-Hessian approximation used by XGBoost [8], the sample-wise gradient and Hessian for class k are

$$\mathbf{G}[t, k, r] = \mathbf{P}[t, k, r] - \mathbf{Y}[k, r], \ \mathbf{H}[t, k, r] = \mathbf{P}[t, k, r](1 - \mathbf{P}[t, k, r]),$$

where $\mathbf{Y}$ is the one-hot label matrix. These values are then aggregated over the samples reaching each node and used to compute the split gain and leaf values.

Therefore, to prove the correctness of $\mathbf{G}$ and $\mathbf{H}$, the prover first proves the correctness of $\mathbf{S}$ and $\mathbf{P}$ in a manner similar to the case of plain GBDT, and then proves the correctness of the above formulas for $\mathbf{G}$ and $\mathbf{H}$. After this, the rest of the proof follows the same structure as the GBDT proof: the prover proves the histogram transformation from sample-wise gradients and Hessians to per-node frequency statistics, propagates the histograms to internal nodes, proves the XGBoost gain for each candidate split, and proves that the selected split is optimal.

F.2 Extending to Random Forests

Statement. The public statement fixes a quantized random-forest trainer, including the number of trees, tree-shape policy, split objective, leaf-output rule, and quantization parameters. The prover commits to the dataset, labels, forest, leaf values, and the bootstrap multiplicities. The verifier should be convinced that the committed forest is obtained by running this public random-forest trainer on the committed data.

Plaintext trainer. Unlike GBDT and XGBoost, random forests [5] have no boosting state: the trees are trained independently, and there is no round-by-round score tensor such as $\mathbf{S}$. Each tree directly handles all K classes.

Fix a tree q. The public trainer first fixes a sample multiplicity vector

$$\boldsymbol{\rho}^{(q)} \in \mathbb{F}^M,$$

where $\boldsymbol{\rho}^{(q)}[r]$ records how many times sample r is included in the bootstrap sample of tree q. If sampling without replacement, then $\boldsymbol{\rho}^{(q)}$ is simply a boolean mask.

For classification, fix a node u of tree q, and let I_u be the samples reaching it. For a candidate split specified by feature j and threshold bin b, define

$$L_{u,j,b} := \{\, r \in I_u : \mathbf{D}[r, j] < b \,\}, \tag{F.5}$$

$$R_{u,j,b} := \{\, r \in I_u : \mathbf{D}[r, j] \geq b \,\}. \tag{F.6}$$

For any sample set $A \subseteq [M]$ and class $c \in [K]$, write

$$G_{A,c} := \sum_{r \in A} \boldsymbol{\rho}^{(q)}[r] \cdot \mathbf{Y}[c, r], \qquad G_A := \sum_{c \in [K]} G_{A,c}.$$

Define

$$G_{u,c} = G_{I_u,c}, \qquad G_{L,c} = G_{L_{u,j,b},c}, \qquad G_{R,c} = G_{R_{u,j,b},c},$$

and

$$G_u = G_{I_u}, \qquad G_L = G_{L_{u,j,b}}, \qquad G_R = G_{R_{u,j,b}}.$$

A convenient score for the Gini criterion [6] is

$$\Phi_{\mathsf{rf}}(u, j, b) = \frac{\sum_{c \in [K]} G_{L,c}^2}{G_L} + \frac{\sum_{c \in [K]} G_{R,c}^2}{G_R}, \tag{F.7}$$

which is equivalent, for a fixed parent node, to maximizing the usual Gini gain. The trainer selects the optimal valid candidate; if no valid candidate exists, the node becomes a leaf. The leaf output is either the majority label or the probability vector of class frequencies in the leaf. Inference is by majority vote or by summing the leaf probability vectors across trees and taking the argmax.

Proof sketch. The proof follows almost identically to the GBDT case, i.e., by first proving inference over each tree, then proving the histograms over leaves and internal nodes, then proving the Gini score for each candidate split, and finally proving that the selected split is optimal. The main difference is that we need to maintain a $\mathbf{G} \in \mathbb{F}^{N_{\text{tot}} \times d \times B \times K}$, which is one dimension higher than the $\mathbf{G}$ in the GBDT proof. Nevertheless, the proof of the histogram over leaves is almost identical, and the propagation to internal nodes can be proved by

$$\mathbf{G}[u, j, b, c] = \mathbf{G}[\mathbf{lc}[u], j, b, c] + \mathbf{G}[\mathbf{rc}[u], j, b, c]. \tag{F.8}$$

The proof of the prefix sum over $\mathbf{G}$ is then handled similarly. Finally, the proof of leaf output (which can be a majority label or class-frequency vector) can be handled by committing the output as a (one-hot or probability) vector and proving its correctness with a maximum or argmax proof.

G Deferred Constructions

G.1 Batched Table Lookup

Given content $\mathbf{F}_i \in \mathbb{F}^{n \times m_i}$ and table $\mathbf{T}_i \in \mathbb{F}^{N \times m_i}$ for $i \in [L]$, and commitments to their corresponding polynomial encodings $F_i(X)$ over $\mathbb{H}_{nm_i}$ and $T_i(X)$ over $\mathbb{H}_{Nm_i}$, our goal is to prove

$$\mathbf{F}_0 \| \cdots \mathbf{F}_{L-1} \subseteq \mathbf{T}_0 \| \cdots \mathbf{T}_{L-1},$$

where notation "$\subseteq$" means that each row of the left matrix appears as a row of the right matrix.

The key challenge here is that the table widths m_i vary by nature. For example, consider a one-row matrix (a, b). Suppose it is committed as $f(X)$ over $\mathbb{H}_{1\times 2}$, its corresponding vector over $\mathbb{H}_{1\times 4}$ is

$$\mathbf{f}' = (a, \square, b, \square).$$

Note that $f(\omega_4^0) = f(\omega_2^0) = a$ and $f(\omega_4^2) = f(\omega_2^1) = b$. However, there is no guarantee what values $f(\omega_4^1)$ and $f(\omega_4^3)$ will be.

Our insight is hence to create a mask

$$\mathbf{m} = (1, 0, 1, 0),$$

Hence

$$f(X)m(X)$$

honestly commits to $(a, 0, b, 0)$ over $\mathbb{H}_4$. By Lemma D.3, $m(X) = \lambda_2^{(0)}(X^2)$.

Formally, let $m = \max_i m_i$. We pad each matrix $\mathbf{F}_i$ to $\widehat{\mathbf{F}}$ of width m by

$$\widehat{F}(X) = F(X) \cdot \lambda_{m/m_i}^{(0)}(X^{nm_i}), \tag{G.1}$$

$$\text{where } \lambda_{m/m_i}^{(0)}(X) = \frac{X^{m/m_i} - 1}{m/m_i(X-1)}. \tag{G.2}$$

And we pad each table $\mathbf{T}_i$ to $\widehat{\mathbf{T}}_i$ of width m similarly.

Now that the contents and the tables are aligned, we can combine them into a single table lookup by a verifier challenge α:

$$\sum_i \alpha^i \cdot \widehat{\mathbf{F}}_i \subseteq \sum_i \alpha^i \cdot \widehat{\mathbf{T}}_i. \tag{G.3}$$

The challenge is sampled after the commitments are fixed. If the original relation is false, fix a query row r^* absent from the table. A compressed match with any fixed table row requires a nonzero polynomial in α of degree at most $L-1$ to vanish. A union bound over the N table rows gives

$$\epsilon_{\mathsf{cmp}}(L, N) \leq \frac{N(L-1)}{p}.$$

The compressed relation is then proved using the table lookup argument of [7].

We note that this construction also supports preprocessing, at the cost of computing cached quotients [11] for all $\widehat{\mathbf{T}}_i$ in $O(Lm \log m)$.

G.2 Column-Replication

Given a vector $\mathbf{f} \in \mathbb{F}^n$ and its committed polynomial encoding $f(X)$ and a positive integer m, we want to prove that committed polynomial $G(X)$ encodes the matrix

$$\mathbf{G} = (\mathbf{f} \,\|\, \mathbf{f} \,\|\, \cdots \,\|\, \mathbf{f}) \in \mathbb{F}^{n \times m}.$$

Unlike Lemma D.3, we do not know if there is a closed-form expression for $G(X)$ in terms of $f(X)$. Below we give a direct proof of correctness of $G(X)$.

We prove the following two identities:

$$\forall i \in [n], j \in [m-1].\ \mathbf{G}[i, j+1] - \mathbf{G}[i, j] = 0, \tag{G.4}$$

$$\forall i \in [n].\ \mathbf{G}[i, 0] = \mathbf{f}[i], \tag{G.5}$$

which are proven separately by

$$\exists q_1(X).\ G(X) - G(\omega_{nm}^{-1} X) = q_1(X) \cdot \frac{\nu_{nm}(X)}{\nu_n(X)}, \tag{G.6}$$

$$\exists q_2(X).\ G(X) - f(X) = q_2(X) \cdot \nu_n(X), \tag{G.7}$$

i.e. two zero-checks. The first identity checks all non-first-column positions, while the second binds the first column to $\mathbf{f}$.

G.3 Concrete Constructions for Segmented Prefix Sum

We now give the concrete constructions deferred from Appendix E.1. Throughout, we write $[t] := \{0, 1, \ldots, t-1\}$. For a tensor $\mathbf{F} \in \mathbb{F}^{a \times b \times c}$, we need the following three relations.

Target constraints. Given a commitment to $\mathbf{F}$, prove commitments to $\mathbf{G}$ and $\mathbf{H}$ such that:

1. **Prefix along the last dimension.**

$$\mathbf{G}[i,j,k] = \sum_{k'=0}^{k} \mathbf{F}[i,j,k'], \quad \mathbf{H}[i,j] = \sum_{k' \in [c]} \mathbf{F}[i,j,k'].$$

2. **Prefix along the middle dimension.**

$$\mathbf{G}[i,j,k] = \sum_{j'=0}^{j} \mathbf{F}[i,j',k], \quad \mathbf{H}[i,k] = \sum_{j' \in [b]} \mathbf{F}[i,j',k].$$

3. **Prefix along the first dimension.**

$$\mathbf{G}[i,j,k] = \sum_{i'=0}^{i} \mathbf{F}[i',j,k], \quad \mathbf{H}[j,k] = \sum_{i' \in [a]} \mathbf{F}[i',j,k].$$

We note that the first and third types of prefix sums are actually prefix sums along the row and column directions of a matrix, respectively. For notational simplicity, we write matrix prefix sums for these two types in the following.

Matrix row sum / row-wise prefix sum. Let $\mathbf{F} \in \mathbb{F}^{n \times m}$ be the committed matrix, and let $\mathbf{H} \in \mathbb{F}^{n}$ be the claimed row sum, i.e.

$$\mathbf{H}[i] = \sum_{j \in [m]} \mathbf{F}[i,j]. \tag{G.8}$$

To prove this relation, we first extend $\mathbf{H}$ to a matrix $\widehat{\mathbf{H}} \in \mathbb{F}^{n \times m}$ by padding zeros before the last column:

$$\widehat{\mathbf{H}} = (\mathbf{0} \,\|\, \mathbf{H}) \in \mathbb{F}^{n \times m}. \tag{G.9}$$

Let

$$\mathbb{H}_n^{\text{last}} := \omega_{nm}^{m-1} \mathbb{H}_n$$

be the last-column subdomain, and let $\nu_{\mathbb{H}_n^{\text{last}}}(X)$ denote its vanishing polynomial. The relation between $\mathbf{H}$ and $\widehat{\mathbf{H}}$ is enforced by proving

$$\exists q_1(X).\ \widehat{H}(X) = q_1(X) \cdot \frac{\nu_{nm}(X)}{\nu_{\mathbb{H}_n^{\text{last}}}(X)}, \tag{G.10}$$

$$\exists q_2(X).\ \widehat{H}(X) - H(\omega_{nm}^{-(m-1)} X) = q_2(X) \cdot \nu_{\mathbb{H}_n^{\text{last}}}(X), \tag{G.11}$$

which are reduced to two zero-checks. Intuitively, the first condition says that $\widehat{\mathbf{H}}$ is zero outside the designated positions, while the second says that on those positions it agrees with $\mathbf{H}$ after shifting the last-column domain back to $\mathbb{H}_n$.

Next, we commit to an auxiliary witness $\mathbf{B} \in \mathbb{F}^{n\times m}$, defined as the row-wise prefix sum of $\mathbf{F} - \widehat{\mathbf{H}}$:

$$\mathbf{B}[i,j] := \sum_{k=0}^{j} \big(\mathbf{F}[i,k] - \widehat{\mathbf{H}}[i,k]\big).$$

We then prove the following two identities:

$$\exists q_3(X).\ B(X) = q_3(X) \cdot \nu_{\mathbb{H}_n^{\text{last}}}(X), \tag{G.12}$$

$$B(X) - B(\omega_{nm}^{-1} X) = F(X) - \widehat{H}(X). \tag{G.13}$$

The first identity guarantees that $\mathbf{H}$ is indeed the correct row sum, while the second guarantees that $\mathbf{B}$ is the correct prefix sum witness.

The key observation is that $\mathbf{B}$ is the de facto prefix sum of $\mathbf{F} - \widehat{\mathbf{H}}$:

$$\mathbf{B}[i,j] = \underbrace{\mathbf{B}[i-1 \bmod n, m-1]}_{=0} + \sum_{k=0}^{j} \big(\mathbf{F}[i,k] - \widehat{\mathbf{H}}[i,k]\big). \tag{G.14}$$

Hence $\mathbf{B}[i, m-1] = 0$ implies

$$\sum_{k\in[m]} \mathbf{F}[i,k] = \mathbf{H}[i]. \tag{G.15}$$

To obtain the full row-wise prefix relation, note that

$$\mathbf{G} = \mathbf{B} + \widehat{\mathbf{H}}. \tag{G.16}$$

The second type of prefix sum. We now describe how to prove the second type relation

$$\mathbf{G}[i,j,k] = \sum_{j'=0}^{j} \mathbf{F}[i,j',k], \qquad \mathbf{H}[i,k] = \sum_{j'\in[m]} \mathbf{F}[i,j',k], \tag{G.17}$$

where $\mathbf{F} \in \mathbb{F}^{n\times m\times \ell}$, $\mathbf{G} \in \mathbb{F}^{n\times m\times \ell}$, and $\mathbf{H} \in \mathbb{F}^{n\times \ell}$. The difficulty is that, under the row-major flattening of $\mathbf{F}$, the entries contributing to one fixed $\mathbf{H}[i,k]$ are not stored contiguously: they appear once every ℓ positions inside the i-th block of length $m\ell$.

Step 1: construct a mask. We first commit to a mask matrix

$$\mathbf{M} \in \mathbb{F}^{n\times m\times \ell}, \tag{G.18}$$

such that

$$\mathbf{M}_{\{n\times m\times \ell\}}[i,j,k] = \begin{cases} 1, & j = m-1, \\ 0, & \text{otherwise.} \end{cases} \tag{G.19}$$

Thus, within each block of m rows corresponding to a fixed i, the mask is nonzero exactly on the last row. Equivalently, the support of $\mathbf{M}$ marks the positions at which the full sum over the middle dimension should appear.

To prove that $\mathbf{M}$ is correct, we prove

$$M(X) - M(\omega_{nm\ell}^{-1} X) = \lambda_{m\ell}^{(0)}(X^n \omega_{nm}^{m-1}) - \lambda_{m\ell}^{(0)}(X^n). \tag{G.20}$$

We also prove

$$\mathbf{M} \odot (\mathbf{M} - \mathbf{1}) = \mathbf{0}, \qquad \mathbf{M}[0, m-1, 0] = 1. \tag{G.21}$$

The right-hand side is explicit and can be computed by the verifier. These constraints uniquely characterize the desired periodic mask.

Step 2: construct an index witness. Next, we commit to an index matrix

$$\mathbf{I} \in \mathbb{F}^{n\times m\times \ell}, \tag{G.22}$$

defined as the (vector) prefix sum of the mask multiplied by the mask:

$$\mathbf{I} = \mathbf{I}' \odot \mathbf{M}, \tag{G.23}$$

$$\text{where } \mathbf{I}'_{\{nm\ell\}}[i] = \sum_{i'=0}^{i} \mathbf{M}_{\{nm\ell\}}[i']. \tag{G.24}$$

$\mathbf{I}$ will be used as an indicator of the positions where the sum should be copied to.

Step 3: lift $\mathbf{H}$ *to the shape of* $\mathbf{F}$. We then commit to an aligned tensor

$$\widehat{\mathbf{H}} \in \mathbb{F}^{n\times m\times \ell},$$

such that

$$\widehat{\mathbf{H}}[i,j,k] = \begin{cases} \mathbf{H}[i,k], & j = m-1, \\ 0, & \text{otherwise.} \end{cases}$$

In other words, $\widehat{\mathbf{H}}$ stores each $\mathbf{H}[i,k]$ exactly at the positions selected by the mask, and is zero elsewhere.

The relation between $\mathbf{H}$ and $\widehat{\mathbf{H}}$ is proved through a copy constraint guided by the index witness $\mathbf{I}$. Concretely, after flattening $\mathbf{H}$ as a vector of length $n\ell$, we prove

$$(1, 2, \ldots, n\ell)^\top \,\|\, \mathbf{H} \subseteq \mathbf{I} \,\|\, \widehat{\mathbf{H}}, \tag{G.25}$$

and

$$(\mathbf{1} - \mathbf{M}) \odot \widehat{\mathbf{H}} = \mathbf{0}. \tag{G.26}$$

where $\subseteq$ denotes a lookup. Intuitively, this states that whenever $\mathbf{I}$ indicates the t-th target location, the value written in $\widehat{\mathbf{H}}$ at that location is exactly the t-th entry of $\mathbf{H}$.

Step 4: prove the sum by a column-prefix argument. Finally, we consider

$$\mathbf{R} := \mathbf{F} - \widehat{\mathbf{H}}.$$

We prove that the column-wise prefix sum of $\mathbf{R}$ vanishes at all marked positions. More precisely, let

$$\mathbf{P}_{\{nm\times\ell\}}[i,j] := \sum_{i'=0}^{i} \mathbf{R}_{\{nm\times\ell\}}[i',j]. \tag{G.27}$$

The correctness of $\mathbf{P}$ follows from the third type of prefix sum.

We show that $\mathbf{P}$ vanishes at all positions where $\mathbf{M}$ is nonzero, i.e.

$$\mathbf{P} \odot \mathbf{M} = \mathbf{0}. \tag{G.28}$$

This proves the correctness of $\mathbf{H}$. To prove the correctness of $\mathbf{G}$, we can simply prove $\mathbf{G} = \mathbf{P} + \widehat{\mathbf{H}}$.

Soundness and completeness. Soundness follows from the soundness of the primitives. To see the completeness, note that by applying the copy constraint (Equation (G.25)), we enforce that the only nonzero term in $\widehat{\mathbf{H}}$ is $\mathbf{H}[i,k]$ at $j = m-1$, so Equation (G.27) is equivalent to

$$\sum_{j'=0}^{m-1} \mathbf{F}[i,j',k] - \mathbf{H}[i,k] = 0,$$

namely,

$$\mathbf{H}[i,k] = \sum_{j\in[m]} \mathbf{F}[i,j,k].$$

Example G.1. Consider $\mathbf{F} \in \mathbb{F}^{2\times3\times4}$. After flattening it as a matrix of size 6×4, the entries of $\mathbf{H} \in \mathbb{F}^{2\times4}$ should appear at the last row of each block of three rows. Thus the mask has the form

$$\mathbf{M} = \begin{pmatrix} 0&0&0&0\\ 0&0&0&0\\ 1&1&1&1\\ 0&0&0&0\\ 0&0&0&0\\ 1&1&1&1 \end{pmatrix}.$$

Over $\mathbb{H}_{nm\ell}$, $M(X) - M(\omega_{nm\ell}^{-1}X)$ encodes

$$\begin{pmatrix} -1&0&0&0\\ 0&0&0&0\\ 1&0&0&0\\ -1&0&0&0\\ 0&0&0&0\\ 1&0&0&0 \end{pmatrix} = \begin{pmatrix} 0&0&0&0\\ 0&0&0&0\\ 1&0&0&0\\ 0&0&0&0\\ 0&0&0&0\\ 1&0&0&0 \end{pmatrix} - \begin{pmatrix} 1&0&0&0\\ 0&0&0&0\\ 0&0&0&0\\ 1&0&0&0\\ 0&0&0&0\\ 0&0&0&0 \end{pmatrix}$$

The $\mathbf{I}$ and $\mathbf{I}'$ used in the proof are respectively

$$\mathbf{I} = \begin{pmatrix} 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ 1\,2\,3\,4 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ 5\,6\,7\,8 \end{pmatrix}, \quad \mathbf{I}' = \begin{pmatrix} 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ 1\,2\,3\,4 \\ 4\,4\,4\,4 \\ 4\,4\,4\,4 \\ 5\,6\,7\,8 \end{pmatrix}.$$

H Forest Semantics and Unique Decoding

This section formalizes the plaintext object certified by the well-formedness proof.

Definition H.1 (Quantized regression tree). *Let $\mathcal{X} \coloneqq [B]^d = \{0, \ldots, B-1\}^d$. A depth-$h$ quantized regression tree is a tuple*

$$\mathcal{T} = (V, r, h, V_{\text{int}}, V_{\text{leaf}}, \mathsf{lc}, \mathsf{rc}, e, \theta, w)$$

where V is a finite node set, $r \in V$ is the root, $V = V_{\text{int}} \dot{\cup} V_{\text{leaf}}$, and

$$\mathsf{lc}, \mathsf{rc} : V_{\text{int}} \to V, \qquad e : V_{\text{int}} \to [d],$$
$$\theta : V_{\text{int}} \to [B], \quad w : V_{\text{leaf}} \to \mathbb{F}.$$

The edges $\{(u, \mathsf{lc}(u)), (u, \mathsf{rc}(u)) : u \in V_{\text{int}}\}$ form a rooted full binary tree with root r and distinct children. Writing $\delta(u) \coloneqq 1 + \text{dist}(r, u)$, we require $\delta(u) \in \{1, \ldots, h\}$ for every $u \in V$.

Define the routing map $\rho_{\mathcal{T}} : \mathcal{X} \to \bigcup_{m=1}^{h} V^m$ as follows. For a sample $\mathbf{x} \in \mathcal{X}$, let $\rho_{\mathcal{T}}(\mathbf{x}) = (u_0, \ldots, u_{m-1})$, where $u_0 = r$, $u_{m-1} \in V_{\text{leaf}}$, and for $s \in [m-1]$,

$$u_{s+1} = \begin{cases} \mathsf{lc}(u_s), & \mathbf{x}[e(u_s)] < \theta(u_s), \\ \mathsf{rc}(u_s), & \mathbf{x}[e(u_s)] \geq \theta(u_s). \end{cases}$$

The route terminates at a unique $u_{m-1} \in V_{\text{leaf}}$ with $m \leq h$, and $\mathcal{T} : \mathcal{X} \to \mathbb{F}$ is defined by $\mathcal{T}(\mathbf{x}) = w(u_{m-1})$. The region of a node is $\mathcal{H}_u^{\mathcal{T}} \coloneqq \{\mathbf{x} \in \mathcal{X} : u \in \rho_{\mathcal{T}}(\mathbf{x})\}$; hence $\mathcal{H}_r^{\mathcal{T}} = \mathcal{X}$ and

$$\mathcal{H}_{\mathsf{lc}(u)}^{\mathcal{T}} = \{\mathbf{x} \in \mathcal{H}_u^{\mathcal{T}} : \mathbf{x}[e(u)] < \theta(u)\},$$
$$\mathcal{H}_{\mathsf{rc}(u)}^{\mathcal{T}} = \{\mathbf{x} \in \mathcal{H}_u^{\mathcal{T}} : \mathbf{x}[e(u)] \geq \theta(u)\}.$$

We require $\mathcal{H}_u^{\mathcal{T}} \neq \varnothing$ for every node u.

Definition H.2 (Quantized GBDT forest). *A T-round, K-class quantized GBDT forest is an ordered collection*

$$\mathcal{F} = \left(\mathcal{T}^{(t,k)}\right)_{t \in [T],\, k \in [K]}$$

of TK depth-h quantized regression trees. Here $t \in [T]$ and $k \in [K]$ are zero-based; committed identifier 0 is dummy, and $\mathcal{T}^{(t,k)}$ has identifier $tK + k + 1 \in$

$\{1, \ldots, TK\}$. *Taking leaf scores to include the public learning-rate factor, the forest defines* $\mathsf{Score}_{\mathcal{F}} : \mathcal{X} \to \mathbb{F}^K$ *by*

$$\mathsf{Score}_{\mathcal{F}}(\mathbf{x})[k] = \sum_{t \in [T]} \mathcal{T}^{(t,k)}(\mathbf{x}).$$

For a committed assignment, write

$$\begin{aligned}\mathbf{A}_{\mathcal{F}} \coloneqq (&\mathbf{lc}, \mathbf{rc}, \mathbf{par}, \mathbf{N}^{\leftarrow}, \mathbf{N}^{\rightarrow}, \mathbf{E},\\ &\mathbf{tree}, \mathbf{root}, \mathbf{depth}, \mathbf{is_leaf}, \mathbf{valid}, \mathbf{score}).\end{aligned}$$

Proposition H.3 (Unique forest decoding). *Suppose that* $\mathbf{A}_{\mathcal{F}}$ *satisfies all conditions in Section 5.3. Then it uniquely defines a labeled quantized GBDT forest* $\mathcal{F}$ *with node identifiers in* $\{1, \ldots, N_{\text{int}} + N_{\text{leaf}} - 1\}$. *For each tree, its leaf hyper-boxes form a disjoint partition of* $[B]^d$*; hence every input reaches exactly one leaf and* $\mathsf{Score}_{\mathcal{F}}$ *is well defined.*

Proof. Let

$$\begin{aligned}V &= \{u : \mathbf{valid}[u] = 1\},\\ V_{\text{leaf}} &= \{u \in V : \mathbf{is_leaf}[u] = 1\},\\ V_{\text{int}} &= V \setminus V_{\text{leaf}}.\end{aligned}$$

For every $u \in V_{\text{int}}$, add the edges $u \to \mathbf{lc}[u]$ and $u \to \mathbf{rc}[u]$. By Equations (5.4), (5.19) and (5.20), both children lie in V and are distinct; by Equation (5.22), exactly the nodes in V_{int} have children. Conversely, Equation (5.17) makes every valid non-root node a child of its declared parent, while Equations (5.10) and (5.22) ensure that this parent is valid and internal. Equation (5.16) makes that parent unique.

By Equations (5.3) and (5.21), depth strictly increases along each edge, so the graph is acyclic and each parent chain ends at depth 1. Equations (5.14), (5.17) and (5.18) identify these nodes exactly as roots, and Equations (5.4), (5.13) and (5.15) give one root for each tree identifier in $\{1, \ldots, TK\}$. Finally, Equation (5.6) bounds every root-to-leaf path by h. Collecting these facts, for every tree identifier $\tau \in \{1, \ldots, TK\}$:

1. there is exactly one root r_τ, with $\mathbf{tree}[r_\tau] = \tau$, $\mathbf{depth}[r_\tau] = 1$, and $\mathbf{par}[r_\tau] = 0$;
2. every internal node has two distinct valid children in the same tree, whose unique parent is that node and whose depth is one larger;
3. every valid non-root node has exactly one valid internal parent; and
4. every edge increases depth, every leaf has no children, and every maximal downward path terminates at a leaf by depth at most h.

Thus these committed topology vectors uniquely fix TK legal rooted full-binary-tree topologies. For each such τ, the tree has root r_τ and child maps $\mathsf{lc}(u) = \mathbf{lc}[u]$

and $\mathsf{rc}(u) = \mathbf{rc}[u]$ for every internal node u. For each decoded tree $\mathcal{T}$ and each of its nodes u, let

$$\mathcal{H}_u^{\mathcal{T}} = \prod_{j \in [d]} [\mathbf{N}^{\leftarrow}[u, j], \mathbf{N}^{\rightarrow}[u, j]).$$

Root boxes equal $[0, B)^d$ by Equation (5.23).

For every internal node u, Equations (5.30) and (5.31) define $e(u)$ as the unique feature in $[d]$ satisfying $\mathbf{E}[u, e(u)] = 1$. Define

$$\theta(u) \coloneqq \mathbf{N}^{\rightarrow}[\mathsf{lc}(u), e(u)] = \mathbf{N}^{\leftarrow}[\mathsf{rc}(u), e(u)].$$

For each leaf ℓ, define $w(\ell) = \mathbf{score}[\ell]$. Equations (5.25) to (5.29) imply

$$\mathcal{H}_{\mathsf{lc}(u)}^{\mathcal{T}} = \{\mathbf{x} \in \mathcal{H}_u^{\mathcal{T}} : \mathbf{x}[e(u)] < \theta(u)\},$$
$$\mathcal{H}_{\mathsf{rc}(u)}^{\mathcal{T}} = \{\mathbf{x} \in \mathcal{H}_u^{\mathcal{T}} : \mathbf{x}[e(u)] \geq \theta(u)\}.$$

Equation (5.24) places $\theta(u)$ strictly inside the corresponding parent interval. Hence the two child boxes are disjoint and their union is the parent box. Induction on each subtree shows that its leaf boxes form a disjoint partition of its root box.

The topology, selected features, thresholds, and leaf scores are all recovered from fixed entries of $\mathbf{A}_{\mathcal{F}}$, so no other labeled forest has the same encoding. Together with the leaf-box partition above, this proves unique decoding and well-defined routing, completing the proof.

I Knowledge-Soundness Analysis

This appendix proves Theorem 3.1: from an accepting proof, an extractor recovers a dataset, labels, forest, and training trace satisfying the public relation. The verifier sees only a constant number of group and field elements and checks batched KZG openings and degrees, CQ++ lookups [7], and polynomial identities at verifier-chosen points. The proof therefore first extracts the committed values from these tests and then establishes their GBDT semantics.

I.1 Proof Outline

Fix $\pi \in \{\mathsf{inf}, \mathsf{wf}, \mathsf{tr}\}$, and let $\mathcal{C}_\pi$ be the constraint system imposed by the corresponding protocol in Appendices E.2 to E.4. Its entries are field-level vector identities, domain sums, row-membership relations $\mathbf{F} \subseteq \mathbf{T}$, or histogram-aggregation relations. Commitments and challenges certify these entries but are not part of their semantics. The proof has two steps:

Step 1 (probabilistic). Except with probability ϵ_π, an accepting proof yields degree-bounded polynomials whose evaluation vectors satisfy every entry of $\mathcal{C}_\pi$. This step accounts for commitment and lookup extraction, randomized batching, histogram challenges, and random-point identity tests.

Step 2 (deterministic). Any assignment satisfying $\mathcal{C}_\pi$ has the intended meaning: $\mathcal{C}_{\mathsf{wf}}$ uniquely decodes a forest (Proposition H.3), $\mathcal{C}_{\mathsf{inf}}$ gives correct inference (Lemma I.3), and $\mathcal{C}_{\mathsf{tr}}$ gives exact split selection and a valid execution of the quantized trainer (Proposition I.4 and Lemma I.5).

Step 1 separates cryptographic bookkeeping from the tree and fixed-point arguments of Step 2. After stating the extractable primitives, it verifies three facts:

1. each challenge is sampled after the commitments on which its bad event depends (Appendix I.3);
2. every tested identity and committed quotient has a sufficient degree bound (Appendix I.4); and
3. every randomized test is charged exactly once (Appendix I.5).

Domain lifting reduces all zero claims to one batch and all sum claims to another, while interleaving is deterministic. The number of claims per batch is fixed by the protocol; the data dimensions affect only their domains, all bounded by 2^L. Consequently,

$$\epsilon_\pi \le \epsilon_{\text{commitments}} + \epsilon_{\text{lookups}} + (Q_{\mathsf{FS}} + 1)c_\pi 2^L/p,$$

where c_π is configuration-dependent but witness-independent. Section I.6 gives the precise bound, and the remaining subsections prove Step 2 and the complete security theorem.

I.2 Extractable Primitives

Step 1 uses two extractable components. In the AGM [12], the KZG extractor returns for each commitment a degree-bounded polynomial consistent with all accepted single and batched openings. If ϵ_{KZG} is the error per commitment and N^π_{comm} is the number of commitments, a union bound contributes $N^\pi_{\mathsf{comm}}\epsilon_{\mathsf{KZG}}$, including the batched openings and degree checks.

For CQ++, let $\epsilon_{\mathsf{CQ++}}(Q, N)$ be the knowledge-soundness error for query length Q and table length N, and let N^π_{lookup} be the number of instances. Its extractor returns the row assignments witnessing every accepted membership claim. Since all padded query and table domains have size at most 2^L, lookup extraction contributes at most $N^\pi_{\mathsf{lookup}}\epsilon_{\mathsf{CQ++}}(2^L, 2^L)$.

Range, maximum, argmax, fixed-point division, and quantization checks introduce no additional primitive: they compile into lookups and algebraic identities using limb decompositions, masked comparisons, and committed quotient–remainder witnesses. It remains only to bound the probability that a false algebraic claim survives its random test.

I.3 Binding Challenge

A Fiat–Shamir challenge binds a claim only after the transcript fixes every polynomial on which that claim depends. The schedule is:

Table 14. Degree effects. Here $n \mid 2^L$ is a claim domain, $N \le 2^L$ is the common lifted domain, and c is the maximum constraint degree.

Object	Degree bound
Vector encoding over $\mathbb{H}_n$	$< n \le 2^L$
Degree-c virtual constraint	$< cn$
Constraint lifted to $\mathbb{H}_N$	$< (c-1)n + N \le cN$
Batched zero- or sum-check side	$\le cN - 1$
Honest quotient of that identity	$< (c-1)N$
Interleaved polynomial	$\le \deg f + n$
Replicated encoding on $\mathbb{H}_{nr}$	$< nr \le 2^L$

1. commit to the input, witness, mask, index, and table polynomials;
2. sample the lookup-compression and histogram challenges, then commit to the induced auxiliary polynomials, including the power tables;
3. sample the zero- and sum-check batching challenges η_0 and η_Σ, then commit to their quotient and remainder polynomials;
4. sample the evaluation points z_0 and z_Σ; and
5. batch the required KZG openings.

Thus each challenge is sampled after the transcript fixes the polynomial in its bad event. If a nonzero polynomial A of degree at most D is fixed before $z \leftarrow \mathbb{F}$ is sampled, then

$$\epsilon_{\mathsf{eval}}(D) \coloneqq \Pr[A(z) = 0 \mid A \neq 0] \le \frac{D}{p}. \tag{I.1}$$

The error in proving the value $A(z)$ is already included in ϵ_{KZG}; (I.1) charges only the random choice of z.

The batched zero- and sum-checks test divisibility through

$$A(X) = F_{\mathrm{bat}}(X) - q(X)\nu_N(X),$$

where F_{bat} is the batched virtual constraint and q is the committed quotient. If the claim is false, $\nu_N \nmid F_{\mathrm{bat}}$, so $A \neq 0$ for every q. A bound on the committable degree of q is therefore essential both to the divisibility test and to its soundness bound.

I.4 Degrees of the Tested Identities

The evaluation error D/p and the required quotient degree follow from the degree effects of virtual constraints, domain lifting, interleaving, and replication summarized in Table 14.

Each witness column over $\mathbb{H}_n$ has degree below $n \le 2^L$. The constraints in Appendices E.2 to E.4 have maximum total degree

$$c = 4. \tag{I.2}$$

Table 15. Challenges and soundness charges for protocol π.

Challenge	Failure event	Degree	Charge
η_0 (one per protocol)	nonzero residuals cancel	$m_{\pi,0} - 1$	$(m_{\pi,0} - 1)/p$
η_Σ (one per protocol)	false sums cancel	$m_{\pi,\Sigma} - 1$	$(m_{\pi,\Sigma} - 1)/p$
α (one per histogram)	unequal histograms agree	address space -1	$\le 2^L/p$
compression (two per lookup)	distinct rows compress equally	$\le 2^L$	$\le 2^L/p$ each
z_0, z_Σ (one per identity)	nonzero identity vanishes	$< 4 \cdot 2^L$	$< 4 \cdot 2^L/p$ each
CQ++ internal	invalid lookup accepted	internal	$\epsilon_{\mathsf{CQ++}}$
KZG batch challenge	inconsistent opening accepted	internal	ϵ_{KZG}

Degree four is attained by (E.34), (E.122), (E.123), and (E.120).[11] All other identities have degree at most three.

With $N \le 2^L$ and $c = 4$, an honest quotient has degree below $(c-1)2^L = 3 \cdot 2^L$. If the verifier enforces $\deg q < (c-1)2^L$, the tested polynomial satisfies

$$\deg A \le \max\{cN - 1,\ \deg q + N\} < c \cdot 2^L = 4 \cdot 2^L, \tag{I.3}$$

so (I.1) charges each evaluation point $4 \cdot 2^L/p$. If one relies instead on the SRS bound $d_{\max}$, this changes to $\deg A \le d_{\max} + 2^L$, still $O(2^L)$. The lookup argument's additional SRS requirements are internal to CQ++ and are included in $\epsilon_{\mathsf{CQ++}}$.

I.5 Counting Challenges

Table 15 enumerates every external challenge family, its failure event, the degree of that event as a polynomial in the challenge, and its charge. The last two rows are internal to the extractable primitives and were charged above. Here $m_{\pi,0}$ and $m_{\pi,\Sigma}$ count the zero and sum claims in their batches, H_π counts histogram identities, N^π_{lookup} counts lookup instances, and E_π counts independently evaluated lifted identities. These counters are fixed by the protocol, not the data dimensions.

The batching polynomials in η_0 and η_Σ have degrees one less than their numbers of claims. A histogram failure polynomial has degree one less than its address space; the training address space is $N_{\mathrm{tot}}dB$, and every such space is at most 2^L. Lookup compression has two stages: interleaving-family separation and the column compression of Appendix G.1. Each is charged $2^L/p$, provided the padded table, including the constant block factor, has at most 2^L rows. The evaluation rows use (I.3); CQ++ and KZG retain their primitive errors.

Interleaving itself adds no challenge because the merged vector is a deterministic function of its inputs, with openings derived from existing commitments and public selectors. Root uniqueness in Equation (5.15) is likewise implemented by deterministic routing and a zero check. A challenge-based implementation would instead add one histogram row and a TK/p charge.

Let $\epsilon^\pi_{\mathsf{rbr}}$ denote the total round-by-round error from external randomized tests, H_π the number of histogram identities, E_π the number of independently

[11] Here $\mathbf{internal} = \mathbf{valid} \odot (\mathbf{1} - \mathbf{is_leaf})$ is a virtual polynomial.

evaluated lifted identities, and $N^{\pi}_{\mathsf{lookup}}$ the number of lookup instances. Bounding each batching term by $2^L/p$ gives

$$\epsilon^{\pi}_{\mathsf{rbr}} \leq \frac{c_\pi 2^L}{p}, \tag{I.4}$$

where

$$c_\pi \coloneqq (m_{\pi,0} - 1) + \max\{m_{\pi,\Sigma} - 1, 0\} + H_\pi + 4E_\pi + 2N^{\pi}_{\mathsf{lookup}}. \tag{I.5}$$

The terms charge, respectively, the zero-check batch, sum-check batch, histograms, evaluated identities, and the two compression stages per lookup. The CQ++ and KZG errors remain separate. Thus c_π is almost a constant[12]; T, K, M, d, B, and N_{tot} enter through 2^L.

Size of the constant. An asymptotic analysis does not depend on c_π. Nevertheless, it can't hurt to check its concrete values, which are recorded during our benchmark. Across the soundness benchmark, the counters range over

π	$m_{\pi,0}$	$m_{\pi,\Sigma}$	H_π	$N^{\pi}_{\mathsf{lookup}}$	E_π
inf	32	0	0	4	1
wf	90–93	5	3	7–8	2
tr	262–274	11	6	22–27	2

(I.6)

Thus (I.5) gives $c_{\mathsf{inf}} = 43$, $c_{\mathsf{wf}} \in [118, 123]$, and $c_{\mathsf{tr}} \in [329, 351]$. As also mentioned earlier in footnote, the variation comes from the range-check. In our implementation, to check $x \in [0, \mathcal{B})$, the prover decomposes it into base-2^w limbs by committing them, the number of which is determined by $\mathcal{B}$ and w. As $\mathcal{B}$ grows due to heavier training workload, the number of PCS and lookups grows. Note also that the training counters include the well-formedness checks because a Terrae training proof also contains a well-formedness proof. The range reflects configuration-dependent batching of some training zero-check and lookup instances.

I.6 Assembling Step 1: Round-by-Round Soundness and Fiat–Shamir

Fiat–Shamir challenges are adaptive: after seeing a challenge, the prover may choose later commitments or retry with another transcript prefix. We therefore apply the invariant of Appendix I.3 round by round rather than fixing all bad events in advance.

Call a transcript state *doomed* if its fixed polynomials cannot be extended to witnesses satisfying $\mathcal{C}_\pi$. Conditioned on the preceding transcript, a challenge in Table 15 can move a doomed state to a non-doomed state only by making its fixed, nonzero failure polynomial vanish. The probability of such event is bounded

[12] c_π is affect by $\mathcal{B}$ and w, which in range-check varies the number of auxiliary PCS. The range of this variation, $\lceil \lceil \log \mathcal{B} \rceil / w \rceil$, is rather minor.

by Equation (I.4), and a union bound gives (I.4). Applying the Fiat–Shamir state-restoration reduction once [4], for at most Q_{FS} random-oracle queries, gives

$$\begin{aligned}\epsilon_\pi \le N^\pi_{\mathsf{comm}}\epsilon_{\mathsf{KZG}} + N^\pi_{\mathsf{lookup}}\epsilon_{\mathsf{CQ{+}{+}}}(2^L, 2^L) \\ + (Q_{\mathsf{FS}}+1)\frac{c_\pi 2^L}{p}.\end{aligned} \tag{I.7}$$

Thus every external randomized test is counted once in $\epsilon^\pi_{\mathsf{rbr}}$, and the Fiat–Shamir factor is applied once to their union.

Lemma I.1 (Extraction of the accepted proofs). *For each $\pi \in \{\mathsf{inf}, \mathsf{wf}, \mathsf{tr}\}$ and every PPT prover producing an accepting proof, a polynomial-time extractor, except with probability bounded by* (I.7)*, outputs the degree-bounded polynomials underlying all witness commitments and openings and witnesses for every accepted lookup claim. Their evaluation vectors satisfy all entries of $\mathcal{C}_\pi$.*

Proof. The polynomial-commitment extractor returns the committed polynomials and accepted opening values, except with the first term of (I.7); CQ++ knowledge soundness returns the lookup and range witnesses, except with the second. Condition on both events. Every other entry of $\mathcal{C}_\pi$ is certified by a zero- or sum-check batch, a histogram weighted-sum identity, or lookup compression. By Appendices I.3 and I.4, each failure polynomial is fixed before its challenge and has the bounded degree. A violated entry can therefore survive only through a doomed-to-non-doomed transition. Equation (I.4) bounds their union, and one Fiat–Shamir reduction gives the third term of (I.7).

Corollary I.2 (Asymptotic soundness). *Assume negligible primitive errors. Specifically, ϵ_{KZG} and $\epsilon_{\mathsf{CQ{+}{+}}}(2^L, 2^L)$ are negligible in the security parameter λ, $p \ge 2^{\Omega(\lambda)}$, and 2^L and Q_{FS} are polynomial in λ. Then $\epsilon_\pi = \mathsf{negl}(\lambda)$ for every $\pi \in \{\mathsf{inf}, \mathsf{wf}, \mathsf{tr}\}$.*

Proof. The protocol counters are constant, so the first two terms of (I.7) are constant multiples of negligible quantities, while the third is $\mathrm{poly}(\lambda)/2^{\Omega(\lambda)}$.

Data dimensions affect (I.7) only through 2^L; instance growth matters only if it increases the global domain bound. The statistical condition (I.7) and the arithmetic no-wrap condition (2.11) are independent and must both hold.

Step 2 is deterministic: assume extracted field elements satisfy $\mathcal{C}_\pi$. Range lookups select canonical representatives in the code space of Section 2.4, and (2.11) keeps every constraint expression below $p/2$ in magnitude. Hence vanishing in $\mathbb{F}_p$ is equivalent to vanishing over $\mathbb{Z}$, and a slack check certifies an integer inequality. The inference and training arguments also assume an accepting well-formedness proof for the same forest, so Proposition H.3 applies, as in our evaluation.

I.7 Step 2: Inference

The leaf lookup identifies a leaf of the correct tree whose box contains the query; well-formedness makes it the unique routed leaf. Score aggregation then links the final coordinate comparison to the forest score of Definition H.2.

Lemma I.3 (Inference semantics). *Condition on the successful-extraction event of Lemma I.1, and suppose that the same committed forest has an accepting well-formedness proof. For every public input* $\mathbf{I}[i]$*, the extracted witness selects the unique leaf reached by* $\mathbf{I}[i]$ *in each decoded tree, and the claimed one-hot label* $\mathbf{Y}[\cdot, i]$ *selects a maximum coordinate of* $\mathsf{Score}_{\mathcal{F}}(\mathbf{I}[i])$*.*

Proof. By Proposition H.3, each decoded tree has leaf boxes partitioning $[B]^d$. For each (t, k, i), the leaf lookup fixes a leaf and its score, while the interval constraints place $\mathbf{I}[i]$ in that leaf's box. The partition makes it the unique routed leaf and its fetched score the leaf value. Thus score aggregation gives

$$\mathbf{S}[k, i] = \mathsf{Score}_{\mathcal{F}}(\mathbf{I}[i])[k].$$

The one-hot check on $\mathbf{Y}$, the masked-score identity, and the final range checks yield $\mathbf{S}[k, i] \leq \sum_{k'} \mathbf{Y}[k', i]\mathbf{S}[k', i]$ for every k. Therefore the coordinate selected by $\mathbf{Y}[\cdot, i]$ is a maximum.

I.8 Step 2: Split Selection and Stopping

We next establish the soundness of split selection and stopping, conditioned on successful extraction.

Proposition I.4 (Soundness of split selection and stopping). *Condition on the successful-extraction event of Lemma I.1. Call* (u, j, b) structurally valid *if both induced children are nonempty and* eligible *if, in addition, its quantized gain meets the public threshold* τ_q*. For every valid node* u*, the extracted witness satisfies:*

1. $\boldsymbol{\chi}[u, j, b] = 1$ *exactly for structurally valid candidates, and* $\boldsymbol{\zeta}[u, j, b] = 1$ *exactly for eligible candidates;*
2. *every internal node selects a unique eligible candidate;*
3. *every leaf is at depth* h *or has no eligible candidate; and*
4. *the selected candidate maximizes the fixed-point encoding of* Φ *over all structurally valid candidates, uses the smallest flattened index among ties, and is the split encoded by the committed forest.*

Proof. Fix a valid node u and suppress the round and class indices.

Validity mask. The public table in Equation (E.99) has row $(0, 0)$ and rows $(a, 1)$ for $a \in [M]$. Its lookups make $\boldsymbol{\chi}_{<}[u, j, b]$ and $\boldsymbol{\chi}_{\geq}[u, j, b]$ the exact nonzero indicators of the two candidate counts. Their product in Equation (E.100) is therefore the exact structural-validity mask $\boldsymbol{\chi}[u, j, b]$.

Candidate scores. For a structurally valid candidate, both denominators in Equations (E.101) and (E.102) are positive. Together with Equation (E.106), these identities uniquely determine $\boldsymbol{\Gamma}_{<}$ and $\boldsymbol{\Gamma}_{\geq}$, whose sum is the trainer's fixed-point encoding $\boldsymbol{\Gamma}[u, j, b]$ of $\Phi(u, j, b)$. By (E.110) and (E.111), the leaf constraints similarly determine $\boldsymbol{\Gamma}_{\text{leaf}}[u]$. The conclusions hold over $\mathbb{Z}$ by (2.11).

Eligibility mask. Booleanity and Equation (E.112) force $\boldsymbol{\zeta} \leq \boldsymbol{\chi}$. Where $\boldsymbol{\chi} = 1$, exactly one of the Boolean values $\boldsymbol{\zeta}$ and $\boldsymbol{\chi} - \boldsymbol{\zeta}$ is one. The comparisons in

Equations (E.113) and (E.114) distinguish gains meeting τ_q from those strictly below it, so $\boldsymbol{\zeta}$ is exact.

Internal nodes. If u is internal, the Boolean one-hot vector $\mathbf{T}$ selects unique (j^*, b^*). Equations (E.117) and (E.118) give $\boldsymbol{\Gamma}^*[u] = \boldsymbol{\Gamma}[u, j^*, b^*]$ and $\boldsymbol{\zeta}[u, j^*, b^*] = 1$. In Equation (E.120), the index term has magnitude below dB; its range check therefore makes $\boldsymbol{\Gamma}^*[u]$ at least every structurally valid candidate score and, on a tie, makes the selected flattened index at most $jB+b$. Thus the selected candidate is the smallest-index maximizer.

Leaves. If u is a leaf below depth h, then $\mathbf{d}_{<h}[u] = 1$ by Equation (E.12), and Equation (E.119) gives $\sum_{j,b} \boldsymbol{\zeta}[u, j, b] = 0$. Since the summands are Boolean and $dB < p$ by (2.11), the sum cannot wrap, so every summand is zero. Otherwise the same depth-range lookup gives $\mathbf{depth}[u] = h$.

Forest binding. Equation (E.121) fixes the selected feature, and Equation (E.122) equates the selected threshold with the committed child-box boundaries. By Proposition H.3, (j^*, b^*) is exactly the split of the decoded committed forest.

I.9 Step 2: The Training Trace

Proposition I.4 certifies a node once its histograms are fixed. It remains to bind those histograms to the committed data and current pseudo-residuals and to chain the score state across rounds; the latter requires induction over the public round order.

Lemma I.5 (Training semantics). *Condition on the successful-extraction event of Lemma I.1, and suppose that the same committed forest has an accepting well-formedness proof. If the extracted witness satisfies all constraints in Appendix E.4, then the committed forest and training trace realize the public quantized trainer on the committed data.*

Proof. Induct on t, maintaining that the score state and pseudo-residual slice at the start of round t equal those produced by the public trainer after rounds $0, \ldots, t-1$.

Boosting state. The initialization and transition $\mathbf{S} \mapsto \mathbf{S}' \mapsto \mathbf{G}$ follow from Equations (E.77) and (E.82) and the extracted maximum, exponential-table, and fixed-point division constraints, which uniquely determine the shifted softmax and its quantization.

Routing. Equations (E.79) and (E.81) fetch a leaf of the round-t tree and place the sample in its box. The partition from Proposition H.3 makes this the unique routed leaf.

Histograms. Equations (E.83)– (E.86) bind each routed update to the challenge power indexed by its node-feature-bin address. Equations (E.87)–(E.88) then equate the leaf histograms with the aggregates of exactly those updates, fixing every entry of $\mathbf{C}_{\text{leaf}}$ and $\mathbf{V}_{\text{leaf}}$, including zeros at unused addresses. Equations (E.89)– (E.92) determine every internal histogram, and (E.93)–(E.98) uniquely determine the learning-rate-scaled leaf means. The common factor d from the per-feature updates cancels in the ratio.

Tree and induction. These histograms determine all candidate scores, so Proposition I.4 yields exact masks, an eligible maximizing split bound to the forest at every internal node, and a leaf only at depth h or when no eligible candidate exists. The extracted round-t tree is therefore the public trainer's tree for the current pseudo-residuals. Its proved leaf scores re-establish the invariant for round $t+1$, completing the induction through $t = T-1$.

Thus, conditioned on extraction, the committed dataset, forest, and trace form a valid execution of the public quantized trainer. This implication is deterministic; its only failure probability is (I.7).

I.10 The Complete Statement

Combining extraction, unique forest decoding, and the three Step 2 results gives the complete security statement.

Theorem I.6 (Security of the complete protocol). *Let all public parameters be polynomially bounded in the security parameter λ, let the KZG SRS be honestly generated for the advertised degree bounds, work in the AGM [12], and model Fiat–Shamir as a random oracle. Assume the degree-parameterized PDL assumption used by CQ++ [7], the knowledge soundness of its lookup argument, and extractability of the KZG PCS, including batched openings. Also assume the no-wrap condition* (2.11) *and the soundness bound* (I.7)*. Then the inference, forest-well-formedness, and training protocols of* Terrae *satisfy:*

1. ***Completeness.*** *An honest prover with a witness satisfying the corresponding relation can produce an accepting proof.*
2. ***Knowledge soundness of inference.*** *From an accepting inference prover, a polynomial-time extractor recovers a degree-bounded committed forest and an inference witness whose evaluation on the public input batch equals the claimed output. With an accepting well-formedness proof, the forest is the unique valid forest of Proposition H.3.*
3. ***Knowledge soundness of forest well-formedness.*** *From an accepting well-formedness prover, a polynomial-time extractor recovers a forest satisfying all structural constraints in Appendix E.2, which determine the unique valid topology of Proposition H.3.*
4. ***Knowledge soundness of training.*** *From an accepting training prover, a polynomial-time extractor recovers the committed dataset, labels, forest, and trace satisfying the public quantized trainer. Every internal node selects an eligible split, every leaf is at depth h or has no eligible split, the validity and eligibility masks are exact, and the selected split maximizes the quantized encoding of Φ and is bound to the committed forest (Proposition I.4 and Lemma I.5).*

For $\pi \in \{\mathsf{inf}, \mathsf{wf}, \mathsf{tr}\}$, extraction fails with probability at most ϵ_π in (I.7)*. For several jointly verified proofs, a union bound sums their respective ϵ_π values. These properties imply Theorem 3.1.*

Proof. Polynomial commitments, lookup arguments, and algebraic checks are complete. It remains to place the honest auxiliary witnesses in their declared ranges. In training, every root contains all M samples and each selected split has two nonempty children, so every valid node contains a routed sample and no division uses a zero denominator. The admissibility conditions of Section 2.4 place all fixed-point intermediates and comparison slacks in range, while Appendix I.4 makes the quotient and remainder polynomials committable under the advertised SRS.

For knowledge soundness, apply the extractors of Lemma I.1. Conditioned on success, the extracted values satisfy $\mathcal{C}_\pi$; Proposition H.3 gives well-formedness, Lemma I.3 gives inference, and Proposition I.4 and Lemma I.5 give training. The primitive extraction errors and external algebraic events in Table 15 are the only failures, and their union is bounded by (I.7).