PROBABILISTIC AUDITS FOR VERIFIABLE TRAINING AND OUTCOME IMPROVEMENT IN DECENTRALIZED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Decentralized training of large models presents two critical verification challenges: ensuring the training process was executed correctly (process verification) and confirming the resulting model genuinely improved (outcome verification). Existing solutions like zkML are prohibitively expensive, while prior Proof-of-Learning schemes focus only on the process, failing to guarantee that the final model is actually better. We introduce a comprehensive and efficient framework that addresses both challenges through economically-secured probabilistic audits. First, we propose a protocol where Provers commit to each training step, with a small, random fraction of steps being audited by verifier committees, and we derive a tight detection-cost frontier that minimizes verification overhead. Second, we introduce Proof-of-Improvement (PoI), a novel and lightweight evaluation audit that statistically certifies milestone-based gains (e.g., perplexity reduction) on a committed dataset. Empirically, on a QLoRA fine-tuning task, our process audits reduce verification compute by over 95% compared to full replication, and our PoI audits certify model improvements with high statistical power at a minimal cost.

1 INTRODUCTION

Decentralized training of large models raises a fundamental question: *how can a coordinator (or the public) verify that training was executed faithfully and in such a way that the resulting model improved on a committed evaluation?* Purely cryptographic approaches (e.g., zk-ML) offer strong guarantees but remain orders of magnitude too costly at training scale; e.g., even small training rounds exhibit heavy proof-generation overheads Lavin et al. (2024); Bellachia et al. (2025). Proof-of-Learning (PoL) replaces circuits with replayed spot-checks over a logged trajectory Jia et al. (2021), and Proof-of-Sampling (PoSP) couples random verification with penalties Zhang & Wang (2024); Zhao et al. (2024). However, when applied to *decentralized*, *trustless*, LLM-scale training, these approaches often require redundancy in their verification (in order to avoid adversarial effects from the *auditor* side), and hence, either inherit the cost of frequent replays or force delicate incentive/committee trade-offs that can create high entry barriers.

In this work, we focus on *training-time audits that are both economical and ML-relevant*. Building on the observation that auditing a random fraction of steps suffices to achieve a target single-step detection level, we analyze and instantiate a protocol with (i) binding per-step commitments, (ii) *post-commit* randomized audits by small verifier committees, and (iii) explicit incentives. A key outcome is a simple, actionable *detection–cost frontier*: the probability of catching a single forged step, $\delta(1)$, scales linearly with the audited fraction, $\delta(1) = \alpha\, q$, where q captures committee capture and numerical tolerance; the minimal verification cost for a target $\delta^\star$ then follows in closed form. A windowed commit→sample→reveal→audit pipeline ensures $O(\ell)$ liveness with only a constant per-window overhead, and a tight stake bound enforces incentive compatibility. We also extend the mechanism to a multi-trainer regime (DiLoCo/Streaming-DiLoCo style, cf. Douillard et al. (2023; 2025)) where outer-round aggregation is audited and, on failure, sampled inner audits attribute blame Douillard et al. (2023; 2025). All of these pieces are validated empirically on a QLoRA fine-tuning workload (cf. Dettmers et al. (2023)), matching the predicted $\delta(1) = \alpha q$ law and delivering considerable compute savings versus fully redundant PoL.

Beyond verifying *how* training was executed, we introduce a lightweight evaluation and audit track that verifies *what* training achieved. At chosen milestones (e.g., end of a window), the trainer claims an improvement of at least γ in token log-loss (equivalently, perplexity) over a committed baseline and evaluation root. The coordinator then (post-commit) samples n evaluation tokens and assigns a small committee to recompute *forward-only* log-probs for both models; the claim is accepted if a pre-declared one-sided test (or sequential test) confirms improvement at confidence $1 - \alpha_{\text{stat}}$. The resulting detection of false improvement claims factorizes as $\delta_{\text{PoI}} = \delta_{\text{stat}}(n) \cdot q$ (statistical power times committee factor), and the verification cost is linear in both the committee size and the number of evaluated tokens, reusing the same economics and incentives as training-step audits.

1.1 Relationship to prior work.

Zero-Knowledge Machine Learning (zkML) frameworks Lavin et al. (2024); Bellachia et al. (2025) target cryptographic correctness but face prohibitive proof costs at training scale. PoL establishes replay-based verifiability through spot-checks Jia et al. (2021), and PoSP introduces randomized committees with economic penalties Zhang & Wang (2024); Zhao et al. (2024). Our work complements these lines by: (i) deriving a *tight* detection–cost frontier with explicit committee-capture and tolerance factors; (ii) proving pipelined liveness and incentive compatibility under a realistic, committee-voted audit; (iii) extending to multi-trainer training with attribution; and (iv) adding a *Proof-of-Improvement* track that certifies outcome gains on a committed evaluation with statistical confidence. To our knowledge, milestone-level PoI—integrated with probabilistic audits and committee economics—has not been formalized within verifiable training protocols. On the distributed side, our multi-trainer extension is designed to coexist with low-communication training methods, such as DiLoCo and Streaming-DiLoCo Douillard et al. (2023; 2025), providing the missing verification layer. The interested reader can find a more thorough literature review in Appendix A.

1.2 Contributions.

Our presented methodology implements several mechanisms for auditing the veracity and effectiveness of training and fine-tuning in the context of a decentralized and trustless environment. We state our main contributions below:

- **Protocol.** A practical commit→sample→reveal→audit protocol with small verifier committees and binding per-step commitments (inputs, outputs, metadata). This is done on a windowed schedule, which enables concurrency.
- **Detection–cost frontier.** We derive theoretical results for the detection probability, and give the *minimal* verification cost for any target $\delta^\star$, revealing a simple linear frontier that replaces M full replays with $m \ll M$ audited replays.
- **Liveness and incentives.** We bound end-to-end wall-time under windowed auditing and give a tight stake condition that makes honesty strictly dominant; a repeated-interaction variant further reduces collateral.
- **Multi-trainer extension.** An outer-round aggregation audit with sampled inner audits on failure attributes faults to workers, preserving low typical-case cost while enabling slashing at worker granularity.
- **Proof-of-Improvement (PoI).** A drop-in evaluation-audit track that certifies milestone improvements (e.g., perplexity decrease by $\geq \gamma$) on a committed evaluation set; $\delta_{\text{PoI}} = \delta_{\text{stat}}(n) \cdot q$ and verification cost is linear in committee size and audited tokens.
- **Empirical validation and guidance.** On QLoRA fine-tuning, we match the predicted $\delta(1) = \alpha q$ law, achieve 94–98% compute savings versus fully redundant PoL at practical detection levels, and provide exact heat-map thresholds to pick m for target q.

The rest of this work is organized as follows. We present the core methodologies in Section 2. We devote Section 3 to the analysis (with proofs in Appendix C), and experimentally validate our research in Section 4. Lastly, we present some closing remarks and limitations in Section 5.

2 METHOD: PROBABILISTIC AUDITS WITH COMMITTEES

We introduce a protocol for verifiable decentralized training that ensures both the correctness of the training process and the improvement of the model's performance. The framework is composed of three primary components: (i) a core protocol for auditing individual training steps in a single-prover setting, (ii) an extension of it to multi-prover distributed training, and (iii) a novel mechanism for verifying outcome improvement, which we term Proof-of-Improvement (PoI).

2.1 PROTOCOL PRELIMINARIES

Let $\mathcal{P}$ denote the set of protocol participants, consisting of $M_p \in \mathbb{N}_+$ *Provers*, $\{P_1, \ldots, P_{M_p}\}$, and a population of $M_v \in \mathbb{N}_+$ *Verifiers*, $\{V_1, \ldots, V_{M_v}\}$. A given Prover P performs $\ell \in \mathbb{N}_+$ gradient-update steps on model parameters $\theta \in \mathbb{R}^d$. Each step t transforms θ_{t-1} into θ_t via a (possibly randomized) update function:

$$\theta_t \;=\; \mathsf{Update}\big(\theta_{t-1},\, \mathcal{D}_{t-1},\, \mathcal{L}_{t-1},\, \mathcal{H}_{t-1}\big), \qquad t = 1, \ldots, \ell, \tag{1}$$

where $\mathcal{D}_{t-1}$ represents the data (e.g., indices with Merkle proofs), $\mathcal{L}_{t-1}$ is the loss function, and $\mathcal{H}_{t-1}$ contains auxiliary state such as optimizer state, RNG seeds, and environment identifiers. The Prover's per-step compute cost is $C_p > 0$, while a Verifier incurs a cost of $C_v > 0$ to replay a single step.

We assume a trusted *smart contract* (SC) that manages participant stakes (s_p and s_v), provides public randomness (cf. Syta et al. (2017); Choi et al. (2023), for example) , and executes the protocol logic. All authenticated messages are delivered within a known delay bound Δ. We consider a *static* Byzantine adversary $\mathcal{A}$ who, prior to execution, may corrupt up to f_p of the Prover's update rounds and up to f_v Verifiers within any given audit subcommittee of size m.

Remark. *While our primary analysis considers a static adversary, the post-commitment reveal structure provides inherent resilience against certain adaptive strategies. Since committees are selected using public randomness after commitments are locked in, an adversary has a limited window to corrupt the specific verifiers chosen for an audit adaptively.*

2.2 CORE PROTOCOL: PROBABILISTIC TRAINING AUDITS

The core mechanism verifies the integrity of the training process. To ensure unpredictability while maintaining high throughput, the protocol operates in windows of G steps using a post-commitment, pipelined audit structure, are illustrated in Figure 1 (Left) and detailed in Algorithm 1 (in Appendix B).

1. Commit Phase. For each training step t within a window, the Prover executes the update and constructs a cryptographic commitment h_t that is posted to a public ledger. This commitment acts as a tamper-proof record of the claimed state transition. To ensure these commitments are both secure and scalable for large models, we employ a Merkle tree-based approach. Instead of hashing the entire parameter state, the model parameters θ are partitioned into fixed-size shards, and the Prover commits to the Merkle root of these shards. The binding commitment for each step is a constant-size hash constructed from the Merkle roots of the model state *before* (θ_{t-1}) and *after* (θ_t) the update, along with a *step witness* W_t:

$$h_t = H\left(\mathsf{MerkleRoot}(\theta_{t-1}) || \mathsf{MerkleRoot}(\theta_t) || W_t\right). \tag{2}$$

The witness $W_t := \big(I_t, \Pi_t, O_{t-1}, R_t, \Lambda_t, \Xi\big)$ contains all necessary metadata to reproduce the step, such as batch indices (I_t), their Merkle proofs (Π_t), optimizer state (O_{t-1}), RNG seeds (R_t), hyperparameters (Λ_t), and a hash of the execution environment (Ξ). This structure makes the commitment binding to the specific state transition, preventing the Prover from changing their story after the fact (*ex-post* equivocation).

2. Sample & Reveal Phase. After the Prover has committed to all G steps in a window, the smart contract uses a source of public randomness to sample a small fraction, α, of the steps to be audited. Once these steps are chosen, the Prover is required to "open" their commitments by revealing the full

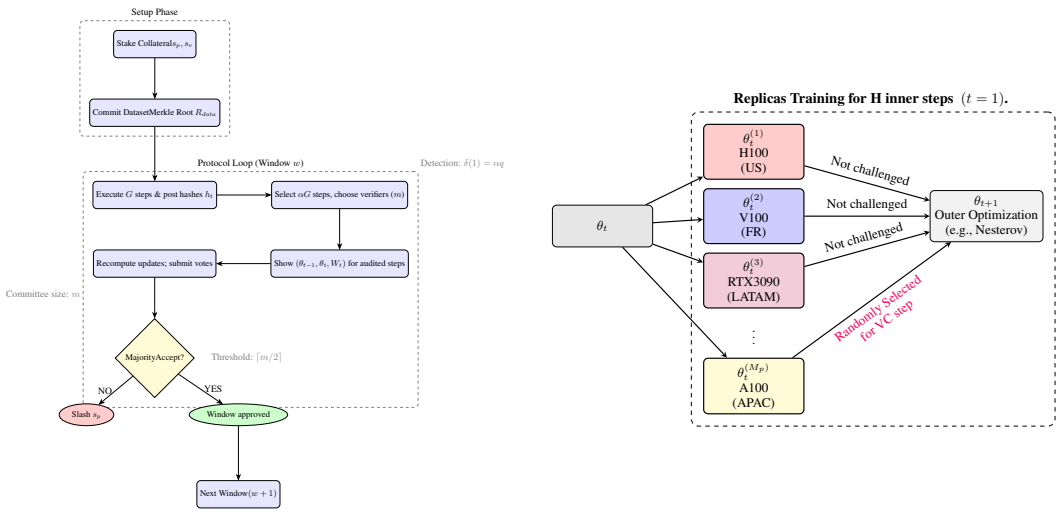

Figure 1: (Left). Proof of learning protocol. (Right). Verifiable DiLoCo. Figure adapted from Douillard et al. (2023)

step witness W_t and the specific parameter shards involved in the update, along with their Merkle proofs. The post-commitment nature of the sampling is crucial; because the Prover does not know which steps will be audited until *after* they have committed to all of them, they are incentivized to perform every step honestly.

3. Audit & Resolve Phase. A random subcommittee of m Verifiers is selected to perform the audit. For each sampled step, the Verifiers first use the revealed Merkle proofs to confirm that the provided parameter shards match the committed roots. Only then do they re-execute the training step using the witness data to produce a recomputed state $\hat{\theta}_t$. They vote to accept the step if their result is sufficiently close to the Prover's claimed result, i.e., $\|\hat{\theta}_t - \theta_t\|_{\mathsf{X}} \leq \tau$, where τ is a small tolerance to account for benign numerical drift across different hardware and $\|\cdot\|_{\mathsf{X}}$ is some appropriate norm. If a supermajority of the committee rejects any step, the Prover's stake is slashed; otherwise, honest Verifiers are rewarded, and the protocol proceeds.

2.3 Extension to Multi-Prover Distributed Training

Our protocol naturally extends to communication-efficient distributed settings like DiLoCo Douillard et al. (2023), where multiple workers (Provers) train in parallel with infrequent synchronization. The process operates in two distinct loops: an *inner loop* of local training and an *outer loop* for global aggregation. Our verification is hierarchical, designed to be efficient in the common case where all workers are honest.

In each outer round r, N_m workers start from a common global model state θ_{r-1}. Each worker i then independently performs k local training steps using its own data:

$$\theta_{r,j}^{(i)} \;=\; \mathsf{Update}\big(\theta_{r,j-1}^{(i)},\, \mathcal{D}_{r,j-1}^{(i)},\, \mathcal{L}_{r,j-1}^{(i)},\, \mathcal{H}_{r,j-1}^{(i)}\big), \qquad t = 1, \ldots, \ell, \tag{3}$$

where $\theta_{r,0}^{(i)} = \theta_{r-1}$. During this phase, each worker acts as a single Prover, creating and posting commitments $(h_{r,j}^{(i)})$ for each of its k local steps, just as described in Section 2.2.

After completing their k local steps, each worker proposes their final local model, $\theta_{r,k}^{(i)}$. The new global model, θ_r, is then computed by aggregating these proposals, for example, through simple averaging: $\theta_r = \frac{1}{N_m} \sum_i \theta_{r,k}^{(i)}$. In more advanced schemes like DiLoCo, this aggregation can also incorporate momentum (e.g., a Nesterov step, cf. Lin et al. (2019)).

Verification proceeds in a two-stage, optimistic fashion to minimize cost:

1. **Stage 1: Aggregation Audit.** First, a verifier committee performs a lightweight check on the outer loop. It verifies that the final global model θ_r was correctly computed from the workers' proposed final models $\theta_{r,k}^{(i)}$. This step is computationally cheap as it only involves re-calculating the aggregation, not re-playing any training steps.
2. **Stage 2: Fault Attribution (Escalation).** *If and only if* the aggregation audit fails, the protocol escalates. A random subset of workers $Q \subseteq \{1, \ldots, N_m\}$ is challenged to reveal their full k-step local training histories. The core probabilistic audit from Section 2.2 is then performed on each challenged worker to find who produced a faulty local model.

This approach is illustrated in Figure 1 (right).

2.4 Evaluation Audit: Proof-of-Improvement (PoI)

To certify that training achieved a meaningful outcome, we introduce PoI, an audit that verifies performance gains at milestones.

At the end of a window, a Prover posts a claim $(r, \gamma, \alpha_{\text{stat}})$:

> "At milestone r, the final model θ_{final} improves token log-loss by at least $\gamma > 0$ (reduces perplexity by a factor $\leq e^{-\gamma}$) versus baseline θ_0 on the committed evaluation set, with one-sided confidence $1 - \alpha_{\text{stat}}$."

This requires a one-time, pre-run commitment to $H(\theta_0)$ and the evaluation data's Merkle root, R_{eval}.

The contract samples a subset $\tilde{S}_r \subseteq D_{\text{eval}}$ of size n. A verifier committee of size m_{eval} computes per-token log-loss differences for $i \in \tilde{S}_r$:

$$Z_i \;:=\; \underbrace{-\log p_{\theta_{\text{final}}}(x_i)}_{\ell_1(x_i)} \;-\; \underbrace{\big(-\log p_{\theta_0}(x_i)\big)}_{\ell_0(x_i)},$$

The claim of improvement is accepted if and only if a pre-declared one-sided statistical test on the sample mean $\bar{Z}$ passes. For example, by confirming that the Lower Confidence Bound (LCB) of the mean improvement meets the claimed threshold:

$$\text{LCB}_{\,1-\alpha_{\text{stat}}}\big(-\bar{Z}\big) \;\geq\; \gamma.$$

This mechanism, pipelined alongside training audits, provides a low-cost, statistically robust method to verify tangible model improvements without requiring monotonic per-step loss.

3 Analysis

We now provide a formal analysis of the protocol introduced in the preceding section. We begin by defining our security and operational goals. Then we prove that our protocol achieves these goals by deriving tight bounds on detection probability, cost, liveness, and incentive compatibility. We present all our proofs in the Appendix C.

We design our protocol to satisfy three fundamental properties:

Definition 1 (Soundness)**.** *Let $\varepsilon_{\text{sound}} \in [0, 1]$. A protocol is* $\varepsilon_{\text{sound}}$-sound *if, against any adversary corrupting up to f_p Prover-rounds and up to f_v verifiers per subcommittee, the probability that any incorrect update is* not *detected is at most $\varepsilon_{\text{sound}}$.*

Definition 2 (Liveness)**.** *A protocol satisfies* liveness *if, when all participants are honest, all ℓ training rounds are completed within a total time of $T \leq \ell\, T_{\text{upd}} + O(\ell/\mathcal{G})$, where T_{upd} is the per-step computation time and $\mathcal{G}$ is the window size.*

Definition 3 (Incentive Compatibility)**.** *Let $\mathcal{G} > 0$ be the Prover's expected gain from a successful one-step cheat, and let $s_p > 0$ be the at-risk stake. The protocol is* incentive-compatible *if the expected utility for cheating is negative, i.e., $U_{\text{cheat}}(f) < U_{\text{honest}} = 0$ for any number of fraudulent steps $f \geq 1$.*

3.1 Soundness and Cost of Process Audits

The soundness of our protocol hinges on the probability that a fraudulent step is both sampled for an audit and correctly flagged by an honest-majority committee.

For any single audited step, the probability q that a forgery is successfully detected is given by:

$$q \;=\; \underbrace{\big(1 - P_{\text{maj-Byz}}(M, F, m)\big)}_{\text{Prob. of honest majority}} \cdot \underbrace{\big(1 - P_{\tau\text{-miss}}\big)}_{\text{Prob. correct flag given honest majority}}, \tag{4}$$

Where $P_{\text{maj-Byz}}$ is the probability of a Byzantine majority in a committee of size m drawn from a population with F adversaries (given by the hypergeometric distribution), and $P_{\tau\text{-miss}}$ is the probability that numerical tolerance τ masks a genuine error. For deterministic computations, $\tau = 0$ and $P_{\tau\text{-miss}} = 0$.

The overall detection probability depends on the number of fraudulent steps f. The base case for a single forgery is linear in the sampling rate α.

Lemma 1 (Base Law for Single Forgery)**.** *For a single forged step ($f = 1$), the detection probability is exactly $\delta(1) = \alpha q$.*

For multiple forgeries, let $X \sim \text{Hypergeom}(\ell, f, \tilde{n})$ be the number of fraudulent steps sampled. The exact detection probability is the expectation over X. Specifically, the probability of detecting *at least* one of f forged steps is:

$$\delta(f) \;=\; 1 - \mathbb{E}\big[(1-q)^X\big] \;=\; 1 - \sum_{k=0}^{\min\{f,\tilde{n}\}} \frac{\binom{f}{k}\binom{\ell-f}{\tilde{n}-k}}{\binom{\ell}{\tilde{n}}} (1-q)^k. \tag{5}$$

While exact, this expression can be tightly bounded for practical design, for instance, by $\delta(f) \geq 1 - (1 - \ alphaq)^f$, which follows from the negative correlation of sampling without replacement (see the Appendix for proof).

3.2 The Cost-Soundness Frontier.

The total expected computational cost is the sum of the Prover's training cost and the expected verification cost.

$$\text{Cost}_{\text{total}}(\alpha, m) \;=\; \ell C_p \;+\; \alpha \ell m C_v. \tag{6}$$

By combining Lemma 1 with the cost model, we arrive at the efficient frontier, which defines the minimum cost to achieve a target soundness level δ^*.

Theorem 1 (Efficient Frontier for Process Audits)**.** *For a target single-step detection probability $\delta^* \in (0, q]$, the minimum achievable cost is:*

$$\text{Cost}_{\min}(\delta^*; m) \;=\; \ell C_p \;+\; \frac{\delta^*}{q}\, \ell m C_v. \tag{7}$$

The equation above establishes a linear tradeoff between verification cost and soundness. Targets where $\delta^* > q$ are infeasible without improving q (e.g., by increasing committee size m).

3.3 Soundness and Cost of Outcome Audits (PoI)

We now present an analogous analysis for PoI. Recall that the goal here is to provide verifiable claims on improvements, rather than on computational work. To that end, notice that the detection probability for a false PoI claim is the product of the statistical power of the test and the committee quality factor:

$$\delta_{\text{PoI}} \;=\; \delta_{\text{stat}}(n) \cdot q_{\text{eval}}. \tag{8}$$

The cost is determined by the number of samples n and the committee size m_{eval}:

$$\text{Cost}_{\text{PoI}}(n, m_{\text{eval}}) \;\approx\; m_{\text{eval}}\, n\big(C_{\text{eval}}(\theta_{\text{final}}) + C_{\text{eval}}(\theta_0)\big). \tag{9}$$

This creates a similar linear cost-soundness tradeoff, where the number of samples n plays a role analogous to the sampling fraction $\alpha\ell$ in process audits. The required sample size n to achieve a desired statistical power can be determined using standard results. For instance, for i.i.d. sub-Gaussian log-loss differences, n scales as $n \gtrsim (\sigma/r)^2$, where r is the margin of the false claim and σ^2 is the variance.

3.4 Pipelined Liveness and Economic Security

Finally, we analyze the protocol's operational guarantees.

Theorem 2 (Pipelined Liveness)**.** *With a window size of $\mathcal{G}$, the total execution time for ℓ steps is bounded by:*

$$T_{\text{total}} \;\le\; \ell\, T_{\text{upd}} \;+\; \left\lceil \frac{\ell}{\mathcal{G}} \right\rceil (2\Delta + \Delta_{\text{aud}}) \;+\; O(1), \tag{10}$$

where T_{upd} is the per-step update time, Δ is the network delay, and Δ_{aud} is the audit finalization time. The pipelined design ensures audit latency contributes only a constant overhead per window, not per step.

The previous theorem demonstrates that the presented protocol achieves *liveness*, loosely speaking, meaning that the cost of the protocol increases moderately compared to the unverified case.

Theorem 3 (Economic Security via Staking)**.** *Honesty is a strictly dominant strategy for a Prover if their stake s_p satisfies:*

$$s_p \;>\; \frac{1-\delta(1)}{\delta(1)}\,\mathcal{G} \;=\; \frac{1-\alpha q}{\alpha q}\,\mathcal{G}, \tag{11}$$

Where $\mathcal{G}$ is the gain from a single successful cheat. An analogous bound holds for PoI claims, replacing $\delta(1)$ with δ_{PoI} and $\mathcal{G}$ with the gain from a false improvement claim, $\mathcal{G}_{claim}$.

4 Experiments

Our experiments test whether the protocol's predictions about *detection*, *cost*, and *liveness* hold in practice, and whether the outcome-audit (PoI) and the distributed (DiLoCo-style) variant behave as the analysis requires. Unless stated otherwise, we fine-tune a Phi-family Abdin et al. (2024) causal LM with LoRA/QLoRA Hu et al. (2022); Dettmers et al. (2023) adapters on WikiText-2 Merity et al. (2016), and run a verifier population M=128 with subcommittee m=7. The theory in Sec. 3 predicts a single-step law $\delta(1) = \alpha\, q$ with $q = (1-P_{\text{maj-Byz}})(1-P_{\tau\text{-miss}})$ and a linear cost–soundness frontier whose intercept and slope are $1/(1+M)$ and $m/(1+M)$ when $C \simeq C_v$. We implement the windowed *commit→sample→reveal→audit* pipeline and the PoI track exactly as analyzed.

Verifying the linear detection law $\delta(1) = \alpha q$. We begin by testing the fundamental prediction that single-step detection scales linearly with the audited fraction. We plant a single forged update at a uniformly random step, draw an independent m-committee for each audited step, and sweep $\alpha \in \{0.05, \ldots, 1.0\}$. The empirical detection curve is a line through the origin whose slope matches the exact q from the hypergeometric committee model, confirming $\delta(1) = \alpha q$ to within binomial uncertainty (Fig. 2 (Left)). This is the base case established in the analysis and underpins all subsequent experiments.

Impact of numerical tolerance Next we isolate how the numeric tolerance τ—used to absorb benign cross-hardware drift—affects the committee factor q. We vary τ, estimate the induced $P_{\tau\text{-miss}}$, and plot $q = (1 - P_{\text{maj-Byz}})(1 - P_{\tau\text{-miss}})$ (Fig. 2 (Middle)). As τ crosses a few multiples of the honest-drift scale, q drops sharply; geometrically, the $\delta(1)$ line rotates downward exactly as predicted by the model. In our deterministic baseline we set τ=0 (hence $P_{\tau\text{-miss}}$=0); for heterogeneous deployments we calibrate τ to a high percentile of observed honest drift on the declared stack Ξ.

Process-audit cost frontier We then examine the verification cost **relative to fully redundant PoL**. Measured normalized cost aligns with the linear frontier that replaces M full replays with $m \ll M$ audited replays: intercept $1/(1+M)$ and slope $m/(1+M)$ when $C \simeq C_v$. Representative operating points lie on the predicted line to plotting precision: for targets $\delta^{=\{0.50,0.80,0.95\}}$ we achieve $\alpha \approx \{0.501, 0.802, 0.952\}$ at normalized costs $\{3.49\%, 5.12\%, 5.94\%\}$ of PoL (Table 1). These values are the ones used later when we compare to wall-clock measurements.

Outcome verification with Proof-of-Improvement We also verify *what* training achieved. At a milestone, the prover claims an improvement $\ge \gamma$ in token log-loss versus a committed baseline; the contract samples n spans and a small evaluation committee runs a one-sided test. On WikiText-2

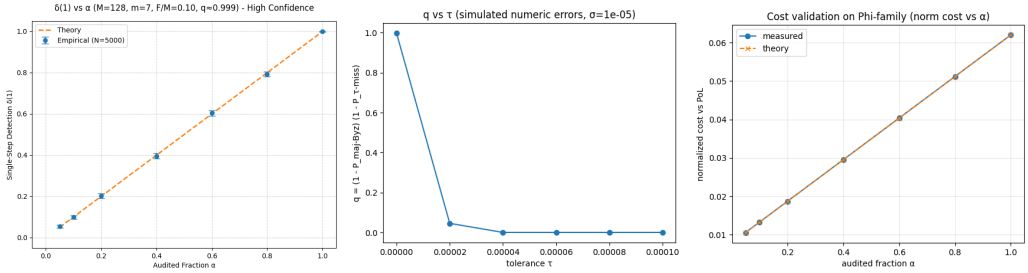

Figure 2: (Left). Single-step detection $\widehat{\delta}(1)$ with 95% CIs vs. α under committee size m=7 and global capture F/M=0.10. Dashed line: $\delta_{\text{theo}}(1) = \alpha\, q$ with q computed exactly from the committee model. (Middle). q as a function of tolerance τ under a numeric-error model; increasing τ reduces q via $P_{\tau\text{-miss}}$. (Right) Normalized cost

δ	required α	normalized cost (%)	comment
0.50	≈ 0.501	≈ 3.49	near half detection at ∼3.5% of PoL
0.80	≈ 0.802	≈ 5.12	∼95% savings vs PoL
0.95	≈ 0.952	≈ 5.94	∼94%+ savings

Table 1: Normalized cost vs. detection target δ (with M=128, m=7).

with a Phi-family LM (LoRA), the full evaluation over 800 spans reports $\Delta_{\text{full}} = 0.4171$ nats/token; sampled audits with $n \in \{50, 100, 200, 400\}$ reject $H_0 : \mu \leq 0$ with very high confidence (e.g., $p = 2.76 \times 10^{-14}$ at n=50). This behavior matches the factorization $\delta_{\text{PoI}} = \delta_{\text{stat}}(n) \cdot q$ and the linear verification cost in $n\, m_{\text{eval}}$ from the analysis.

Cost validation on the Phi family (measured vs. theory) To validate the cost model numerically, we measure prover step time C and verifier replay time C_v on the same Phi-family workload in deterministic mode (F=0 $\Rightarrow$ q=1). The logs show $C_{\text{mean}} = 0.2231$s (median 0.2148s; p95 0.2278s) and $C_{v,\text{mean}} = 0.2149$s (median 0.2134s; p95 0.2243s), confirming the regime $C \simeq C_v$ assumed by the frontier. Using these times with M=128, m=7, the measured normalized cost tracks the theoretical line almost perfectly when plotted against α (Fig. 2 (Left)). The same operating points as in Table 1 are realized by the wall-clock measurements (e.g., $\delta^{=0.80}$ at $\alpha \approx 0.802$ costs $\approx 5.12\%$ of PoL), strengthening external validity of the linear law on real compute.

Auditing distributed training (DiLoCo-style) with attribution Finally, we exercise the two-stage distributed audit: an outer-round aggregation check with sampled inner audits for attribution on failure. In a toy run with N_m=2 workers, k=3 local steps per outer round, R=2 outer rounds, outer audit rate α_{out}=1, inner escalation rate β_{inner}=0.5, and tight tolerance $\tau = 10^{-6}$, we inject both a faulty local step and a faulty aggregation. The outer audit flags a failure and the escalation identifies a faulty worker (`identified_any_faulty=True`), demonstrating end-to-end detection and attribution under the design analyzed in Sec. 3.

Across these six experiments, we find consistent agreement between practice and theory. The single-step law $\delta(1) = \alpha q$ holds; tolerance τ depresses q exactly as predicted; process audits obey a linear cost frontier whose slope/intercept are confirmed by wall-clock timings; PoI delivers statistical power with verification cost linear in n; and the distributed variant detects aggregation failures and attributes blame with low typical-case overhead.

5 Closing Remarks

In this work, we addressed two fundamental challenges in decentralized model training: ensuring the training was executed correctly (process verification) and confirming the resulting model genuinely improved (outcome verification). We introduced a comprehensive framework that combines efficient, economically secured probabilistic audits for training steps with a novel and lightweight evaluation audit we term Proof-of-Improvement (PoI). Our process audits leverage a commit-sample-

target q	F/M=0.05	0.10	0.20	0.30	0.40
≥ 0.99	3	5	11	—	—
≥ 0.95	3	3	7	15	—

Table 2: Minimal odd committee sizes achieving target q for selected global capture F/M. Entries are exact (hypergeometric).

n	$\widehat{\Delta}(n)$	$\mathrm{std}(Z)$	p-value	reject H_0	Δ_{full}	$\lvert\widehat{\Delta} - \Delta_{\text{full}}\rvert$
50	0.4650	0.3164	2.76×10^{-14}	true	0.4171	0.0479
100	0.5029	0.3779	$< 10^{-16}$	true		0.0858
200	0.4988	0.3947	$< 10^{-16}$	true		0.0817
400	0.5027	0.3784	$< 10^{-16}$	true		0.0856

Table 3: One-sided t-test at $\alpha_{\text{stat}} = 0.05$, $\gamma = 0$.

reveal protocol with verifier committees to achieve high security guarantees at a fraction of the cost of exhaustive replay methods. PoI complements this by enabling provers to make statistically verifiable claims about performance gains on a committed dataset, directly tying the verification process to tangible model quality. Our theoretical analysis established a clear and actionable linear trade-off between verification cost and security, encapsulated by the single-step detection law $\delta(1) = \alpha q$ and a minimal cost frontier for achieving any target detection level. These theoretical predictions were validated empirically on a QLoRA fine-tuning task, where our protocol reduced verification compute by over 95% compared to fully redundant PoL while maintaining strong detection guarantees. Together, these contributions offer a practical and deployable path toward verifiable decentralized training that covers both the correctness of the updates and the improvement of the final model, staying within realistic cost budgets for modern LLM workflows.

5.1 Limitations

While our results are promising, we acknowledge certain limitations that open avenues for future research. Our empirical validation was focused on a fine-tuning workload, and our multi-trainer experiment was designed to demonstrate the fault-attribution mechanism rather than operate at a large scale. Extending this evaluation to more complex scenarios like large-scale pre-training is an important next step.

In addition, our analysis focuses primarily on verification compute savings and does not fully quantify the overheads incurred by the Prover. The per-step cryptographic commitment, which involves calculating a Merkle root over the model's entire state, introduces computational costs that scale with model size. Furthermore, the "Reveal" phase for audited steps requires transmitting the step witness, which includes the optimizer state. For optimizers like Adam, this state can be substantial, representing a potential communication bottleneck that merits further investigation, especially in bandwidth-constrained decentralized environments.

Furthermore, we acknowledge that our security analysis primarily considers a static Byzantine adversary. While the post-commitment reveal structure offers some protection, the protocol's resilience against more sophisticated, adaptive adversaries warrants a deeper investigation. Such adversaries, who might attempt to corrupt verifiers after a committee is selected, are countered by the assumption of a short finalization window. Future work should formally analyze the network and smart contract timing assumptions required to secure this window and explore stronger cryptographic mechanisms to mitigate these adaptive risks.

Finally, our implementation of PoI certifies improvement based on log-loss reduction. We believe the PoI framework is generalizable, but extending it to accommodate a broader class of evaluation metrics, particularly complex, non-differentiable, or sequence-level metrics related to safety and alignment, is a key avenue for future research.

REFERENCES

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

Ahmed Ayoub Bellachia, Mouhamed Amine Bouchiha, Yacine Ghamri-Doudane, and Mourad Rabah. Verifbfl: Leveraging zk-SNARKs for a verifiable blockchained federated learning. *arXiv preprint arXiv:2501.04319*, 2025.

Bing-Jyue Chen, Suppakit Waiwitlikhit, Ion Stoica, and Daniel Kang. ZKML: An optimizing system for ML inference in zero-knowledge proofs. In *Proceedings of the Nineteenth European Conference on Computer Systems*, pp. 560–574, 2024.

Kevin Choi, Aathira Manoj, and Joseph Bonneau. Sok: Distributed randomness beacons. In *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 75–92. IEEE, 2023.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Arthur Douillard, Qixuan Feng, Andrei A Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, Marc'Aurelio Ranzato, Arthur Szlam, and Jiajun Shen. DiLoCo: Distributed low-communication training of language models. *arXiv preprint arXiv:2311.08105*, 2023.

Arthur Douillard, Yanislav Donchev, Keith Rush, Satyen Kale, Zachary Charles, Zachary Garrett, Gabriel Teston, Dave Lacey, Ross McIlroy, Jiajun Shen, et al. Streaming DiLoCo with overlapping communication: Towards a distributed free lunch. *arXiv preprint arXiv:2501.18512*, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Meng Chen, Zhifeng Chen, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems*, 2019.

Hengrui Jia, Mohammad Yaghini, Christopher A Choquette-Choo, Natalie Dullerud, Anvith Thudi, Varun Chandrasekaran, and Nicolas Papernot. Proof-of-Learning: Definitions and practice. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 1039–1056. IEEE, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ryan Lavin, Xuekai Liu, Hardhik Mohanty, Logan Norman, Giovanni Zaarour, and Bhaskar Krishnamachari. A survey on the applications of zero-knowledge proofs. *arXiv preprint arXiv:2408.00243*, 2024.

Yann LeCun. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998.

Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. *arXiv preprint arXiv:1908.06281*, 2019.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*, 2018.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. https://arxiv.org/abs/1910.02054, 2020. arXiv:1910.02054.

Alexander Sergeev and Mike Del Balso. Horovod: Fast and easy distributed deep learning in tensorflow. https://arxiv.org/abs/1802.05799, 2018. arXiv:1802.05799.

Mohammad Shoeybi, Mostafa P. Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. https://arxiv.org/abs/1909.08053, 2019. arXiv:1909.08053.

Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J Fischer, and Bryan Ford. Scalable bias-resistant distributed randomness. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 444–460. Ieee, 2017.

Justin Thaler et al. Proofs, Arguments, and Zero-Knowledge. *Foundations and Trends® in Privacy and Security*, 4(2–4):117–660, 2022.

Yue Zhang and Shouqiao Wang. Proof of sampling: A nash equilibrium-secured verification protocol for decentralized systems. *arXiv preprint arXiv:2405.00295*, 2024.

Zishuo Zhao, Zhixuan Fang, Xuechao Wang, Xi Chen, Hongxu Su, Haibo Xiao, and Yuan Zhou. Proof-of-Learning with incentive security. *arXiv preprint arXiv:2404.09005*, 2024.

Appendix

A Literature Review

In what follows, we present a more thorough literature review of the state of the art of verifiable compute and distributed training in the context of decentralized training of large-scale machine learning models.

A.1 Cryptographic Proofs (zkML)

One approach to trustless verification is to use cryptographic proofs, notably zero-knowledge succinct arguments (zk-SNARKs and STARKs Lavin et al. (2024)), to prove that training computations were carried out correctly. In principle, zk-SNARKs can provide a succinct proof of a large computation (like an LLM training step) that anyone can quickly verify on-chain, with the proof size and verify time independent of the model's size Thaler et al. (2022). This gives cryptographic guarantees of *correctness* – a malicious trainer cannot cheat the proof Thaler et al. (2022). In practice, however, compiling a massive neural network training into a SNARK circuit is extremely expensive, with recent analyses suggesting an increase of several orders of magnitude in overhead in computation cost and latency for even just inference tasks in a ZK circuit Chen et al. (2024). A recent framework (VerifBFL Bellachia et al. (2025)) demonstrated verifiable federated learning by generating zk-SNARK proofs for each participant's local training. While the results seem somewhat promising, they are still far from implementable. Indeed, for a relatively simple convolutional neural network trained on the MNIST handwritten dataset LeCun (1998), the authors observed that the on-chain verification was fast (¡0.6 s), but producing a proof for even a tiny training round took on the order of 81 seconds Bellachia et al. (2025). This overhead is prohibitive for large models or many training iterations. Fully zk-proving the training of a 70B-parameter LLM is impractical for the foreseeable future, absent breakthroughs in proof efficiency.

A.2 Proof-of-Learning (PoL)

Introduced by Jia et al. (2021), PoL leverages the fact that training a model (via, e.g., stochastic gradient descent or ADAM Kingma & Ba (2014)) produces a unique trajectory of model updates that is hard to forge without doing the work. In a PoL scheme, the prover (trainer) logs a sequence of intermediate states – e.g., model weights and hyper-parameters after each batch or epoch – along with metadata like batch indices and random seeds. This sequence is the "certificate" of training. A verifier can then randomly spot-check some of these intermediate steps: they pick a random subset of steps and re-compute the training transition (e.g., take the recorded weights at step k, apply the claimed gradient on the stated batch k, and check if it indeed produces the recorded weights at step $k+1$) Jia et al. (2021). If all checked steps are consistent, the verifier gains confidence that the entire sequence (from initial weights to final model) results from legitimate training. By adjusting how many steps are verified, one can trade off verification cost for assurance level. The security argument for PoL is that constructing a fake training log is as hard as training the model – essentially, inverting or short-cutting SGD is difficult. For example, an attacker would have to find gradients that produce a desired final model without actually computing them, which, in general, is computationally as expensive as honest training. A main drawback of PoL is its computational cost. Indeed, it is shown in Jia et al. (2021) that the time complexity of the verification step evolves as

$$\mathsf{Cost}(\texttt{PoL}) = \mathcal{O}(E \cdot Q \cdot \ell \cdot C_{|\theta|}), \tag{12}$$

where $E \in \mathbb{N}$ is the number of epochs in the training algorithm, $Q \in \mathbb{N}$ is the number of verifications per epoch, $\ell \in \mathbb{N}$ is the so-called checkpointing interval (i.e., how often the protocol checkpoints) and $C_{|\theta|} \in \mathbb{R}_+$ is the computational cost associated with training a model as a function of its weights, $\theta \in \mathbb{R}^d$, which is notably large for LLMs. Notice that Equation equation 12 only considers a single verifier, which can, in turn, lend itself to collusion. In a more general setting, one would employ a committee of M verifiers but would need to increase the computational cost in equation 12 by a factor of M, accordingly.

Furthermore, the computational cost of PoL induces a *Verifier's Dilemma*: verifying many steps can be costly, so if not adequately incentivized, verifiers might be lazy and skip checks, undermining security.

In order to accommodate for potential discrepancies arising from, e.g., differences in floating point algebra, Jia et al. (2021) proposes to take proof as valid if the output weights from the provider and the validator $\theta_{\text{end}}^{\text{prover}}, \theta_{\text{end}}^{\text{verifier}}$, respectively, are sufficiently close. More precisely, given some measure of distance $d : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ and some tolerance $\mathsf{Tol} > 0$, a proof is taken as valid in their setting if $d(\theta_{\text{end}}^{\text{prover}}, \theta_{\text{end}}^{\text{verifier}}) < \mathsf{Tol}$.

Recent extensions of PoL incorporate stronger incentive models. For example, in Zhao et al. (2024), the authors propose a "capture-the-flag" game where verifiers earn extra rewards by finding any inconsistency (flags) in the proof, ensuring they check diligently. We intend to explore, improve, and expand these techniques and extend them to create (i) more computationally efficient methodologies and (ii) a base protocol with fully distributed training.

Another variant proposed in Zhang & Wang (2024) is the so-called *Proof-of-Sampling Protocol (PoSP)*. In their model, computations are taken as valid with probability $1-p$ and otherwise verified by a committee of M verifiers with probability p. Should a computation be deemed as *invalid*, the prover gets penalized an amount that is large enough so that, rationally, their best strategy is always to submit a *valid* computation. Put differently, the expected reward from *cheating* is smaller than the expected rewards from performing the computation. Intuitively, this approach incurs a cost of the order of

$$\mathsf{Cost}(\texttt{PoSP}) = \mathcal{O}\left(p \cdot M \cdot E \cdot Q \cdot \ell \cdot C_{|\theta|}\right), \tag{13}$$

which is an improvement over equation 12 provided that $pM < 1$, i.e., if the verification probability satisfies $p < 1/M$. This, in turn, creates a delicate balance. On the one hand, if one cares about fault tolerance, then M must be relatively high, which means that the proportion of verified computations is small. This could, in turn, lead to ill behaviors from the provers or arbitrarily large potential penalties (often expressed as staked amounts), which might result in entry barriers from the protocol. These high entry barriers also imply that only a few providers and verifiers can join the network, which might, in turn, lead to centralization. On the other hand, reducing the size of the committee to its minimum (e.g., $M = 2$) would yield minimal computational gains while at the same time exposing the protocol to verifier collusion (intuitively, the smaller the verifying committee is, the easier it will be to manipulate).

A.3 Decentralized Distributed Training

Training large language models is often an exceedingly expensive computational task that requires computation due to their vast parameter sizes and data-intensive workloads. One common way of alleviating these computational costs is through distributing the computational load. While there is a vast literature on the topic, see, e.g., Sergeev & Del Balso (2018); Shoeybi et al. (2019); Huang et al. (2019); Rajbhandari et al. (2020); Lin et al. (2018) we will focus specifically on methods that allow for distributed training across multiple different machines in different locations. Central to this are the works of Douillard et al., Douillard et al. (2023; 2025) have proposed *Distributed Low-Communication* (DiLoCo) Douillard et al. (2023), a distributed optimization algorithm aimed at drastically reducing communication frequency in LLM training. Instead of synchronizing gradients at every minibatch, DiLoCo performs many local updates on each worker (using ADAM Kingma & Ba (2014) as the local optimizer) before occasionally averaging models across workers, using an outer loop with, e.g., *Nesterov momentum* Lin et al. (2019); Douillard et al. (2023). This approach allows training on "islands" of devices that are only intermittently connected, relaxing the typical requirement of a high-speed interconnect. DiLoCo achieved model quality on par with conventional data-parallel training on a standard large-scale dataset while communicating 500× less frequently among workers Douillard et al. (2023). In practical terms, eight workers communicating only once every 500 training steps matched the accuracy of fully synchronous training, demonstrating that vast reductions in communication are possible without sacrificing convergence. Moreover, DiLoCo was robust to heterogeneous data distributions across workers and resilient to dynamic availability of resources (workers can drop out or join during training with minimal impact) Douillard et al. (2023). Building on this idea, Douillard et al. (2025) introduced an enhanced strategy often referred to as Streaming DiLoCo, aiming to minimize communication overhead and latency penalties

further Douillard et al. (2025). Streaming DiLoCo improves upon the original method by (i) partially synchronizing parameters, significantly reducing the peak bandwidth required at any given time Douillard et al. (2025), (ii) increasing efficiency in the implementation, and (iii) quantizing exchanged model updates to lower precision, cutting down the total volume of data transferred between workers Douillard et al. (2025). By combining these techniques, the authors were able to show that it is possible to distribute training of a billion-parameter transformer and reach similar accuracy as fully synchronous training while reducing required inter-worker bandwidth by about two orders of magnitude (a 100× reduction) Douillard et al. (2025). These low-communication approaches are significant because they enable multi-cluster or geographically distributed training of LLMs without the necessity of dedicated super-computing infrastructure.

B ALGORITHM

Algorithm 1 The Probabilistic Audit Protocol (for one window w)

Input: Window size G, sampling fraction α, committee size m.
1. Commit Phase (Prover)
for $t = wG + 1$ **to** $(w + 1)G$ **do**
  Execute $\theta_t = \mathsf{Update}(\theta_{t-1}, W_t)$ and post commitment h_t to the ledger.
end for
2. Sample Phase (Smart Contract)
After finalization of window commitments, derive public randomness r_w (e.g., via VRF).
Sample a set of audited steps $\tilde{N}_w \subset \{wG{+}1, \ldots, (w{+}1)G\}$ of size $|\tilde{N}_w| = \lceil \alpha G \rceil$ uniformly without replacement.
3. Reveal Phase (Prover)
for each audited $t \in \tilde{N}_w$ **do**
  Reveal the tuple (W_t), the specific parameter shards of θ_{t-1} and θ_t, involved in the update, and their corresponding Merkle proofs against the committed roots.
end for
4. Audit & Vote Phase (Verifier Subcommittee)
for each $t \in \tilde{N}_w$ **do**
  Using r_w, select a random subcommittee of m Verifiers.
  Each member of the subcommittee first uses the provided Merkle proofs to verify that the revealed parameter shards correctly reconstruct the committed Merkle roots for θ_{t-1}, θ_t. Only then they proceed to recompute $\widehat{\theta}_t = \mathsf{Update}(\theta_{t-1}, \mathcal{D}_{t-1}, \mathcal{H}_{t-1})$ and votes **accept** iff $\|\widehat{\theta}_t - \theta_t\|_{\mathsf{X}} \leq \tau$.
end for
5. Resolve Phase (Smart Contract)
if for any $t \in \tilde{N}_w$, the number of **accept** votes is less than $\lceil m/2 \rceil$ **then**
  Slash the Prover's stake s_p and halt the protocol.
else
  Reward Verifiers who voted with the majority and proceed to the next window.
end if

C PROOFS

Lemma 1 (Base Law for Single Forgery) Let $\alpha \in (0, 1]$ be the fraction of audited steps and $q \in (0, 1]$ be the probability that an honest-majority committee correctly identifies a forgery. For a single forged step ($f = 1$), the detection probability is $\delta(1) = \alpha q$.

Proof. Let A be the event that the single forged step is sampled for an audit, and let B be the event that the verifier committee correctly detects the forgery. The sampling is uniform and random, thus $P(A) = \alpha$. The conditional probability of detection, given the step is sampled, is defined as $P(B|A) = q$. Since the sampling event and the committee's verification are independent, the total probability of detection is the joint probability:

$$\delta(1) = P(A \cap B) = P(B|A)P(A) = q\alpha$$

□

Theorem 1 (Efficient Frontier for Process Audits) Let $\ell \in \mathbb{N}_+$ be the total number of training steps, with per-step costs C_p for the Prover and C_v for a Verifier. Let $m \in \mathbb{N}_+$ be the committee size. For a target single-step detection probability $\delta^* \in (0, q]$, the minimum verification cost is given by:

$$\text{Cost}_{\min}(\delta^*; m) = \ell C_p + \frac{\delta^*}{q} \ell m C_v$$

Proof. The total expected cost is the sum of the Prover's computation cost and the expected verification cost:

$$\text{Cost}_{\text{total}}(\alpha, m) = \ell C_p + \mathbb{E}[\text{Verification Cost}] = \ell C_p + (\alpha \ell) m C_v$$

To achieve a target soundness level δ^*, we require $\delta(1) = \delta^*$. By Lemma 1, $\alpha q = \delta^*$, which implies the necessary sampling fraction is $\alpha = \delta^*/q$. Since $\alpha \leq 1$, it must hold that $\delta^* \leq q$. Substituting this expression for α into the cost function yields the minimal cost for the target soundness δ^*:

$$\text{Cost}_{\min}(\delta^*; m) = \ell C_p + \left(\frac{\delta^*}{q}\right) \ell m C_v$$

This establishes a linear trade-off between the verification cost and the soundness guarantee. □

Theorem 2 (Pipelined Liveness) Let $\ell \in \mathbb{N}_+$ be the total number of steps, processed in windows of size $G \in \mathbb{N}_+$. Let T_{upd} be the per-step computation time, Δ be the network delay, and Δ_{aud} be the audit finalization time. The total execution time is bounded by:

$$T_{\text{total}} \leq \ell T_{upd} + \left\lceil \frac{\ell}{G} \right\rceil (2\Delta + \Delta_{aud}) + O(1)$$

Proof. The total time T_{total} is the sum of the Prover's sequential computation time and the cumulative latency from the audit pipeline. The total computation time across all l steps is $\ell \cdot T_{upd}$. The protocol is processed in $N_w = \lceil \ell/G \rceil$ windows. The pipelined design ensures that the latency of auditing window w overlaps with the computation of window $w + 1$. Thus, latency contributes an overhead per window, not per step. The latency for one window consists of at least two network delays (commit finalization, reveal) and the audit time Δ_{aud}. The total cumulative latency is $N_w(2\Delta + \Delta_{aud})$. Combining these terms, we obtain the upper bound on the total time. □

Theorem 3 (Economic Security via Staking) Let $\mathcal{G} > 0$ be the Prover's gain from a single successful cheat and let $s_p > 0$ be the Prover's stake. Honesty is a strictly dominant strategy if:

$$s_p > \left(\frac{1}{\alpha q} - 1\right) \mathcal{G}$$

Proof. Let the utility of honesty be $U_{\text{honest}} = 0$. The expected utility of attempting a single cheat, $\mathbb{E}[U_{\text{cheat}}]$, is determined by the two possible outcomes: success (no detection) or failure (detection). The probability of detection is $\delta(1) = \alpha q$.

$$\mathbb{E}[U_{\text{cheat}}] = P(\text{success}) \cdot (\text{Gain}) + P(\text{failure}) \cdot (\text{Loss})$$

$$\mathbb{E}[U_{\text{cheat}}] = (1 - \delta(1)) \cdot \mathcal{G} + \delta(1) \cdot (-s_p)$$

For honesty to be strictly dominant, we require $\mathbb{E}[U_{\text{cheat}}] < 0$:

$$(1 - \delta(1))\mathcal{G} - \delta(1) s_p < 0 \implies (1 - \delta(1))\mathcal{G} < \delta(1) s_p$$

Solving for the stake s_p yields:

$$s_p > \frac{1 - \delta(1)}{\delta(1)} \mathcal{G}$$

Substituting $\delta(1) = \alpha q$ from Lemma 1 gives the required condition:

$$s_p > \frac{1 - \alpha q}{\alpha q} \mathcal{G} = \left(\frac{1}{\alpha q} - 1\right) G$$

This ensures that the expected utility of cheating is negative, making it an economically irrational strategy. □

D STATEMENT ON THE USE OF AI

In preparing this manuscript, we utilized large language models as a productivity tool. Their assistance was helpful for improving the clarity and tone of the writing, for grammatical and consistency checks, during initial research ideation, and for debugging segments of the experimental code. The final content and all intellectual contributions are the authors' own.