COMMUNICATION EFFICIENT FEDERATED LEARNING VIA DISTRIBUTED BLACK-BOX ZERO ORDER OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated learning (FL) enables collaborative model training while keeping raw client data local, but the repeated exchange of high-dimensional gradients or model updates can make communication the dominant systems cost. Existing communication-efficient FL methods reduce this burden mainly through gradient or model compression, sparsification, or zero-order updates built from random perturbation directions. We propose *BlackBoxFL*, a communication-efficient federated zero-order protocol that takes a different approach: clients perform local black-box optimization over a small number of low-dimensional parameter blocks and communicate only the resulting block candidates, scalar losses, and one-bit control decisions. In each cycle, a small set of blocks is selected at random. Each client then searches the corresponding low-dimensional region using forward-pass loss evaluations alone, combining dyadic narrowing with finite-difference descent, and transmits a candidate only when it improves sufficiently over its local baseline. The server performs only a loss-weighted least-squares combination of the reported candidates, after which clients vote on fresh local data to accept or roll back the aggregate update. Unlike gradient-compression methods, BlackBoxFL therefore communicates no gradients, and unlike seed-based zero-order methods, it communicates no directional estimates derived from random perturbations. For a fixed block size, number of selected blocks, and number of clients, both the uplink and the downlink communication per cycle are independent of the ambient model dimension. We complement the implemented protocol with a certified exact-loss analysis based on Lipschitz global optimization. Under mild continuity and smoothness assumptions, the idealized search over a selected block converges to the global optimum of the corresponding block-restricted objective, and, with repeated block coverage and Lipschitz pruning, the associated fully blockwise method converges to the global optimum of the full objective. Experiments on MNIST and CIFAR-10 image classification, and on character-level language modeling with tiny-Shakespeare, demonstrate substantial savings: on five benchmarks, BlackBoxFL uses 7–25$\times$ less communication than the most communication-efficient of the from-scratch zero-order baselines in our main comparison and roughly two to four orders of magnitude less than the others, while attaining the highest accuracy among zero-order methods across all evaluated tasks. These gains come at the cost of substantially more local forward-pass evaluations.

1 INTRODUCTION

Federated learning (FL) trains a shared model across clients such as smartphones and edge devices while keeping their raw data local (Zhang et al., 2021; Li et al., 2020). Rather than transmitting samples, clients typically send gradients, parameter updates, or compressed model information to a coordinating server. This avoids centralizing the training data, but the repeated exchange of high-dimensional updates can become the dominant systems bottleneck, particularly for large models and bandwidth-limited clients.

Most communication-efficient FL methods follow two complementary directions. *Gradient and model compression* reduces the size of each update through quantization, sparsification, error feedback,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

and momentum. Representative methods include Mem-SGD (Stich et al., 2018), DoubleSqueeze (Tang et al., 2019a), block-wise SignSGD (Zheng et al., 2019; Bernstein et al., 2018), signSGD-FV (Park et al., 2024), FetchSGD (Rothchild et al., 2020), Artemis (Philippenko & Dieuleveut, 2020), CSER (Xie et al., 2020), DoCoFL (Dorfman et al., 2023), and EF21-P (Gruntkowska et al., 2023). *Network sparsification* instead communicates only a subset of model parameters, building on the lottery-ticket view of sparse subnetworks (Frankle & Carbin, 2018; Ramanujan et al., 2020) and its federated instantiations FedPM (Isik et al., 2022) and KLMS (Isik et al., 2024). BiCompFL (Egger et al., 2025) combines uplink and downlink compression, but remains a first-order method because its mask logits are trained by backpropagation; we therefore use it in Appendix L.1 as a compressed first-order reference. Appendix A provides a broader discussion.

Federated zero-order (ZO) optimization removes the need to transmit client gradients or locally trained model updates, relying instead on loss evaluations or finite differences. FedZO (Fang et al., 2022) establishes nonconvex guarantees under partial participation, with a refined analysis in (Chen et al., 2023b); BAFFLE (Feng et al., 2024) eliminates backpropagation through scalar loss differences. DeComFL (Li et al., 2025) and FedKSeed (Qin et al., 2024) regenerate perturbation directions from shared randomness, reducing both uplink and downlink communication to a few scalars independently of the model dimension. FeedSign (Cai et al., 2025) further reduces each update to a shared seed and a sign bit, while HiSo (Li et al., 2026), FZooS (Shu et al., 2023), and Meerkat (Ran et al., 2025) address query efficiency, surrogate-gradient construction, and non-IID drift. These methods primarily communicate directional information obtained from random perturbations. BlackBoxFL instead communicates the outcome of each client's own local search over a low-dimensional parameter block, together with a vote on whether the aggregated result should be kept.

Our approach also draws on Lipschitz global optimization, where the search domain is adaptively partitioned and regions are discarded using certified lower bounds. This lineage includes Piyavskii–Shubert, DIRECT (Jones et al., 1993), DOO/SOO (Munos, 2011), $\mathcal{X}$-armed bandits (Bubeck et al., 2011), and Lipschitz global optimization (Malherbe & Vayatis, 2017); coordinatewise finite differences date to Kiefer–Wolfowitz (Kiefer & Wolfowitz, 1952), and ZO-BCD (Cai et al., 2021) applies zeroth-order block coordinate descent to huge-scale black-box problems in the centralized setting. Fed-PNE (Li et al., 2024) and its personalized variants (Li et al., 2023; Arabzadeh et al., 2024) federate $\mathcal{X}$-armed search with few communication rounds, but maintain and refine a partition of the full search domain. This is appropriate for low-dimensional continuum-armed problems but becomes infeasible in the 10^4–10^6 dimensional parameter spaces of neural networks.

BlackBoxFL retains the partition-based global-search mechanism but applies it only to a small number of low-dimensional coordinate slices, replacing a full-dimensional partition tree by repeated low-dimensional block searches that each client runs locally, with bit-level federated feedback. Related distributed black-box methods (Ito et al., 2019; Ben-Tal et al., 2015; He et al., 2022; Akhavan et al., 2021; Bartan & Pilanci, 2019; Tang et al., 2019b; Rai & Mou, 2023; Tsitsiklis et al., 1986; Nedić & Olshevsky, 2013; Chen et al., 2023a; Mhanna & Assaad, 2023) are discussed in Appendix A.

Main contributions. We introduce *BlackBoxFL*, a communication-efficient federated zero-order protocol that optimizes only a small number of low-dimensional parameter blocks in each cycle. Clients evaluate their private local losses using forward passes; a client transmits a single bit when it declines to report, and a full quantized point and loss otherwise. Blocks are drawn from a random permutation of all blocks, consumed in order and redrawn once exhausted: a blind selection process with no server-side ranking involved. Each active block is searched entirely locally by every client (dyadic narrowing followed by finite-difference gradient descent); a per-client transmit-gate withholds reports that fail to clear a client's own improvement threshold; and the server combines whatever is reported by loss-weighted least squares. Clients then decide by vote, on freshly sampled local data, whether to accept or roll back the result, so the server itself never touches any data. Consequently, with blocks selected from a shared per-epoch seed, both the uplink and the downlink communication per cycle are independent of the ambient model dimension.

We analyze an exact-loss conceptual counterpart based on optimistic space partitioning with smoothness-matched Lipschitz pruning. In this certified method, a conventional block-gradient update is replaced by zero-order global optimization over the selected coordinate slice. Proposition 4.3 shows that each such block search converges to the block-restricted global optimum. Under randomized block selection, the fully blockwise conceptual method converges almost surely to the

global optimum (Corollary 4.5), while its block-coordinate local stage admits a best-iterate rate of $\mathcal{O}(t^{-1/2+\varepsilon})$ for every $\varepsilon \in (0, \frac{1}{2})$ and improves to $\mathcal{O}(1/t)$ under quadratic growth (Theorem B.5). For the full-dimensional local refinement, we additionally obtain a locally linear rate (Theorem B.4); for the synchronous full-dimensional partition scheme, smoothness-matched pruning yields a uniformly bounded number of surviving cells and bounded cumulative suboptimality under an appropriate refinement schedule. These guarantees cover the conceptual method only; Section 4 gives the conditions under which they carry over to the implemented protocol.

2 Problem Formulation

Federated objective and oracle model. Assume that we have a dataset $\mathcal{D}$, partitioned among K clients: $\mathcal{D} = \bigcup_{k=1}^{K} \mathcal{D}_k$, where client k keeps its local dataset $\mathcal{D}_k$ private. For model parameters $\mathbf{x} \in \mathbb{R}^d$, define

$$f_k(\mathbf{x}) := \ell(\mathbf{x} \mid \mathcal{D}_k), \qquad f(\mathbf{x}) := \sum_{k=1}^{K} p_k f_k(\mathbf{x}),$$
$$p_k \geq 0, \qquad \sum_{k=1}^{K} p_k = 1. \tag{1}$$

The weights may represent client dataset sizes or application-specific priorities, as well as statistical properties of the various data sets. Without loss of generality $p_k = 1/K$ (Qu et al., 2021; Collins et al., 2022), since the weights p_k can be incorporated into f_k. We seek

$$f^* := \min_{\mathbf{x} \in \mathcal{C}} f(\mathbf{x}), \qquad X^* := \arg\min_{\mathbf{x} \in \mathcal{C}} f(\mathbf{x}), \tag{2}$$

and write $\mathbf{x}^* \in X^*$ when a particular minimizer is required. Clients access f_k only through forward-pass loss evaluations and, in the implemented protocol, transmit only the prescribed activity bits, per-block points and losses when reporting (or a single decline bit otherwise), and accept/reject votes, never gradients or model updates.

Box-constrained parameter domain. We optimize over a compact axis-aligned box

$$\mathcal{C} = \prod_{i=1}^{d} [c^i_{\min}, c^i_{\max}] \subset \mathbb{R}^d, \qquad c^i_{\min} < c^i_{\max}. \tag{3}$$

This assumption matches the coordinatewise search regions maintained by BlackBoxFL and is natural for projected neural-network training, explicit weight bounds, or a bounded trust region around initialization. The affine change of variables

$$z_i = \frac{x_i - c^i_{\min}}{c^i_{\max} - c^i_{\min}} \tag{4}$$

maps $\mathcal{C}$ bijectively onto $[0,1]^d$. Thus, without loss of generality, the conceptual partition analysis is stated on the unit cube; L_0, L_1, and μ denote the constants in these normalized coordinates. Compactness and continuity guarantee $X^* \neq \emptyset$. To be able to prove global convergence we need the following mild assumption:

Assumption 2.1 (Lipschitz regularity)**.** *Each f_k is continuously differentiable on a neighborhood of $\mathcal{C}$, and there exist $L_0, L_1 > 0$ such that for all $\mathbf{x}, \mathbf{y} \in \mathcal{C}$,*

$$|f_k(\mathbf{x}) - f_k(\mathbf{y})| \leq L_0 \|\mathbf{x} - \mathbf{y}\|_2, \qquad \|\nabla f_k(\mathbf{x}) - \nabla f_k(\mathbf{y})\|_2 \leq L_1 \|\mathbf{x} - \mathbf{y}\|_2.$$

The following assumption is required in convergence rate analysis:

Assumption 2.2 (Local quadratic growth around the global optimum)**.** *There exist $\rho, \mu > 0$ such that for every $\mathbf{x}^* \in X^*$ the ball $\mathbb{B}(\mathbf{x}^*, \rho)$ is contained in $\mathcal{C}$, f is convex on it, and $f(\mathbf{x}) - f(\mathbf{x}^*) \geq \frac{\mu}{2}\|\mathbf{x} - \mathbf{x}^*\|^2$ for all $\mathbf{x} \in \mathbb{B}(\mathbf{x}^*, \rho)$.*

Lemma 2.3 (Finitely many global minimizers)**.** *Under Assumption 2.2, distinct global minimizers are at distance at least ρ from one another. Hence X^* is finite, and in normalized coordinates $g := |X^*| \leq \left(\lfloor \sqrt{d}/\rho \rfloor + 1\right)^d$.*

The proof is given in Appendix D. Assumption 2.2 isolates the global minimizers (Lemma 2.3); its quadratic growth matches the $\mathcal{O}(L_1\delta^2)$ slack of the pruning bound (9), keeping the number of surviving cells bounded (Theorem B.6; for $f(x) = x^4$ the count grows as $\Theta(2^{n/2})$); and it yields the local rates of Theorems B.3–B.5. The consistency results (Propositions 4.2 and 4.4, Corollary 4.5) do not use it. The ReLU networks in our experiments (TinyCNN and LeNet5) are only piecewise smooth, so for them Assumption 2.1 holds within each activation region rather than globally; this does not affect the experiments, and the guarantees of Section 4 concern the conceptual method. For the deterministic conceptual analysis, continuity of each f_k on $\mathcal{C}$ suffices for grid consistency (Proposition 4.2); Assumption 2.1 is invoked only where smoothness is needed, for pruning and rates. The averaged objective inherits the same bounds.

For a coordinate set I and a step $h > 0$, whenever the symmetric probes are feasible, define the coordinatewise central-difference estimator

$$\widetilde{\nabla}_I\varphi(\mathbf{x}; h) := [\widetilde{\nabla}_i\varphi(\mathbf{x}; h)]_{i\in I}^T, \tag{5a}$$

$$\widetilde{\nabla}_i\varphi(\mathbf{x}; h) := \frac{\varphi(\mathbf{x} + h\mathbf{e}_i) - \varphi(\mathbf{x} - h\mathbf{e}_i)}{2h}. \tag{5b}$$

When h is clear, it is omitted. If φ has an L_1-Lipschitz gradient, then

$$|\widetilde{\nabla}_i\varphi(\mathbf{x}) - \nabla_i\varphi(\mathbf{x})| \le \frac{L_1 h}{2}, \tag{6}$$

$$\|\widetilde{\nabla}_I\varphi(\mathbf{x}) - \nabla_I\varphi(\mathbf{x})\|_2 \le \frac{\sqrt{|I|}}{2} L_1 h. \tag{7}$$

(integrate $\nabla_i\varphi$ along $[\mathbf{x} - h\mathbf{e}_i, \mathbf{x} + h\mathbf{e}_i]$). A d'-dimensional block estimate costs $2d'$ forward passes ($2d' + 1$ with the center value). In the full-dimensional partition analysis, a cell of diameter δ_n uses

$$h_n = \vartheta \frac{\delta_n}{\sqrt{d}}, \qquad \vartheta \in (0, \tfrac{1}{2}], \tag{8}$$

which vanishes with the mesh and, since the cell's side is $\delta_n/\sqrt{d}$, keeps both probes $\mathbf{x}_Q \pm h_n\mathbf{e}_i$ of its center inside the cell. Probe feasibility for local refinement is handled in Definition B.2.

Communication measure. Communication is counted in bits per cycle, $\mathcal{N}_B = \mathcal{N}_{\text{up}} + \mathcal{N}_{\text{down}}$; Section 5 bounds the uplink and the downlink separately. Block selection relies on shared randomness:

Assumption 2.4 (Shared randomness)**.** *All clients and the server run the same pseudorandom generator. At the start of each epoch the server broadcasts a seed of n_s bits (we use $n_s = 64$), from which every client regenerates that epoch's partition of the coordinates into blocks and the permutation in which the blocks are visited.*

This is the shared-seed assumption of seed-based federated zero-order methods (Li et al., 2025; Qin et al., 2024; Cai et al., 2025).

3 The BlackBoxFL Protocol

BlackBoxFL is a block-coordinate protocol in which, at each cycle, every client searches a small number of low-dimensional parameter blocks with respect to its own loss and returns an improved local candidate, communicating only its parameters and loss rather than gradients or perturbation-direction estimates. Communication is performed every cycle on a subset of B_{cap} blocks. The coordinates are partitioned tensor by tensor into $B = \lceil d/d' \rceil$ blocks $\{I_b\}_{b=1}^B$ of at most d' coordinates each. An epoch consists of $N_{\text{cyc}} = \lfloor B/B_{\text{cap}} \rfloor$ cycles. Visiting the blocks in a fresh random permutation each epoch is a standard random-reshuffling scheme for block-coordinate methods (Wright, 2015). After the model is exhausted, blocks are reshuffled, and a new epoch begins. (Section K discusses the implementation). At each cycle i, block $b_{t,i}$ is searched within the axis-aligned box $Q_b = [\mathbf{x}_{I_b} - \rho_0, \mathbf{x}_{I_b} + \rho_0]$ around its own coordinates $\mathbf{x}_{I_b}$; the box is rebuilt from scratch every visit, so no search state persists between visits. The partition and the block permutation are redrawn each epoch from the shared seed (Assumption 2.4), so every client knows which blocks each cycle selects and no block identities are transmitted. Gradient-like quantities in both the analysis and the

Algorithm 1 BlackBoxFL: High-Level Block Protocol

Input: Initial model $\mathbf{x}$; blocks $\{I_b\}_{b=1}^{B}$; block budget B_{cap}; clients K.
for epoch $t = 1, 2, \ldots$ **do**
  The server broadcasts a fresh seed; every client derives the epoch's partition and block permutation from it.
  for $i = 1, \ldots, N_{\mathrm{cyc}}$ **do**
    Every client draw the next B_{cap} blocks of the shared permutation.
    for each selected block b **do**
      Screen activity: Each client sets one bit indicating whether its local gradient estimate at b's current value exceeds a threshold.
    end for
    Each client sends its B_{cap} activity bits in one message.
    For each block b if fewer than half the clients report activity, the server skips b for this cycle.
    The server broadcasts which selected blocks are active.
    for each **active** selected block b **do**
      Local search (free): Each client searches its own copy of b's box entirely locally, dyadic narrowing followed by finite-difference gradient descent, committing a move only when it improves on its own last point.
      Transmit-gate: Each client compares its local result against its own pre-search baseline; if the improvement is below a threshold it sets one "nothing to report" bit instead of its full data.
    end for
    All agents report results for all B_{cap} blocks.
    Combine: For each block with at least one report, the server combines all clients' points into one candidate by loss-weighted least squares (WLS), a block without reports stays unchanged.
    The server transmits parameters of WLS estimates for relevant blocks.
    Accept/reject: Every client evaluates, for each block, both the server's candidate and b's pre-update value on the same freshly sampled local batch, and votes one bit per block: yes if the candidate lowers its loss.
    The server updates only the blocks for which at least half the clients vote yes, and broadcasts the outcome for each block; any other block reverts to its original value.
  end for
end for
Output: Final global model $\mathbf{x}$.

implementation use the central-difference estimator (5) defined in Section 2; no backpropagation or model update is communicated, and the server never accesses training or validation data of its own.

The protocol is summarized in Algorithm 1. Complete pseudocode, including all thresholds, step-size schedules, quantization, and the local-search commit rule, is provided in the supplementary material (see Algorithm 2).

The stages differ sharply in cost. Activity screening is the only one every client pays for regardless of outcome. Local search is free: each client narrows and descends on its own copy of the block box without contacting the server. The transmit-gate then reduces that work to either a full report or a single decline bit, which the server combines in closed form by loss-weighted least squares (Appendix C). Because the server holds no data of its own, it cannot check the combined candidate against a validation split; instead, every client re-evaluates it on freshly sampled local data, and the majority vote decides whether to keep it.

Both uplink and downlink are batched per cycle and, for fixed B_{cap}, d', and K, independent of d (Section 5).

4 Global Optimization Guarantees

In this section, we analyze a global optimization algorithm based on division into cubes, using the Lipschitz condition to prune sub-optimal cubes together with a local gradient descent algorithm. We prove that the algorithm converges to the global optimum provided that the loss and its gradient satisfy a Lipschitz condition. BlackBoxFL is motivated by a blockwise coordinate-descent version of this global optimization, since complexity 2^d is unreasonable for the size of typical neural networks; its local search is not itself a certified global optimizer.

We first analyze the exact-loss method, a certified global optimizer that refines a partition of the full feasible box and prunes only cells that cannot contain a global minimizer, and then its block-coordinate realization, which remains globally optimal over each block given the other parameters.

Certified full-dimensional search. The global search is based on division into 2^d sub-blocks, and pruning of sub-blocks based on the Lipschitz condition. We begin with the normalized box $\mathcal{C} = [0,1]^d$ of (4). At stage n, every retained box is bisected in all coordinates, the objective and central finite differences are evaluated at the child centers, and a child is discarded only when a smoothness-based lower bound exceeds the best value observed so far. For a child Q with center $\mathbf{x}_Q$, diameter δ_Q, and finite-difference step h_n, the following bound is used. We refer to discarding a cell only when such a lower bound exceeds the incumbent as *Lipschitz pruning*; by Lemma 4.1, it never removes a cell containing a global minimizer.

Lemma 4.1 (Smoothness-matched Lipschitz pruning)**.** *If f has L_1-Lipschitz gradient, then every $\mathbf{x} \in Q$ satisfies*

$$\begin{aligned} f(\mathbf{x}) &\geq \mathrm{LB}(Q) \\ &:= f(\mathbf{x}_Q) - \left(\|\widetilde{\nabla} f(\mathbf{x}_Q)\| + \frac{\sqrt{d}}{2} L_1 h_n \right) \frac{\delta_Q}{2} - \frac{L_1}{8} \delta_Q^2. \end{aligned} \tag{9}$$

Hence a cell containing a global minimizer is never removed when the algorithm prunes only cells with $\mathrm{LB}(Q)$ larger than the incumbent value. Bounded evaluation or quantization error is handled by adding the margin stated in Appendix G.

Lemma 4.1 requires knowledge of L_1. We note that any upper bound can be used. Furthermore, such a bound can be inferred using the network architecture, and the bounds on the parameters (e.g. based on quantization). All other quantities on the right-hand side of (9), including the finite-difference estimates, are known to the algorithm.

Proposition 4.2 (Grid consistency)**.** *Let $G_m \subset \mathcal{C}$ be finite grids whose covering radii tend to zero, and let $\mathbf{x}_m \in \arg\min_{\mathbf{x} \in G_m} f(\mathbf{x})$. For compact $\mathcal{C}$ and continuous f, $f(\mathbf{x}_m) \to f^* := \min_{\mathcal{C}} f$ and $\mathrm{dist}(\mathbf{x}_m, X^*) \to 0$. If the global minimizer is unique, then $\mathbf{x}_m \to \mathbf{x}^*$.*

Because Lipschitz pruning preserves at least one cell intersecting X^* and the retained mesh tends to zero, Proposition 4.2 implies convergence of the method to the global optimum. The conceptual method is described in Appendix B and the proof is given in Appendix E.

Bisecting one d-dimensional cell creates 2^d children. Thus, even though the number of surviving cells is eventually bounded under Assumptions 2.1 and 2.2, both the transient query cost and the bound itself remain exponential in d. Direct full-dimensional search is therefore infeasible for neural-network parameter spaces. Therefore, we resort to block coordinate optimization. Let I_b be a selected block and keep all complementary coordinates fixed at the current iterate $\mathbf{x}$. Define

$$\mathcal{C}_b(\mathbf{x}) := \{\mathbf{v} : (\mathbf{v}; \mathbf{x}_{-I_b}) \in \mathcal{C}\}, \qquad f_b(\mathbf{v}; \mathbf{x}) := f(\mathbf{v}; \mathbf{x}_{-I_b}). \tag{10}$$

The ideal block update replaces a single block-gradient step by

$$\mathbf{x}^+_{I_b} \in \operatorname*{arg\,min}_{\mathbf{v} \in \mathcal{C}_b(\mathbf{x})} f_b(\mathbf{v}; \mathbf{x}), \qquad \mathbf{x}^+_{-I_b} = \mathbf{x}_{-I_b}.$$

Proposition 4.3 (Slice-global optimization)**.** *For every nonempty coordinate slice $\mathcal{C}_b(\mathbf{x})$, the restricted objective inherits the continuity and smoothness assumptions of f. Consequently, the certified partition search in dimension $|I_b| \leq d'$ drives its best sampled value to $\min_{\mathbf{v} \in \mathcal{C}_b(\mathbf{x})} f_b(\mathbf{v}; \mathbf{x})$.*

Unlike a block-gradient step, this search can move to a distant point of lower loss on the selected slice, escaping points that are only locally stable; it cannot escape points that require several blocks to change jointly, so slice-global optimality alone does not imply joint global optimality.

Proposition 4.4 (Randomized block coverage)**.** *Let $N_i(n)$ count the refinements of coordinate i by stage n, where the partition of the coordinates into blocks may change from stage to stage. If at every stage each coordinate is refined with probability at least $p > 0$, independently across stages, evaluations are exact, and cells are removed only by Lipschitz pruning, then almost surely $N_i(n) \to \infty$ for every i and, in normalized coordinates,*

$$\delta_n^2 \leq \sum_{i=1}^{d} 4^{-N_i(n)} \longrightarrow 0.$$

Accordingly, the best sampled value converges to f^*. Furthermore, assuming a unique global optimum, the best sampled point converges to $\mathbf{x}^*$.

Proposition 4.4 supplies the missing joint coverage: although each individual operation is low-dimensional, every coordinate direction is refined infinitely often. Combining Lipschitz pruning, grid consistency, and positive-probability block coverage gives the main global guarantee (proof in Appendix I).

Corollary 4.5 (Global convergence of the fully blockwise conceptual method)**.** *For the exact-loss conceptual method with Lipschitz pruning and randomized block selection satisfying Proposition 4.4, the best sampled value converges to f^* almost surely. If the global minimizer is unique, the best sampled point converges to it.*

Corollary 4.5 is proved in Appendix J. The guarantee is asymptotic and should be read as a consistency result rather than an iteration-complexity bound; in practice BlackBoxFL runs a fixed block budget per cycle for finitely many epochs, which our experiments show suffices for strong accuracy at low communication.

To accelerate convergence, the conceptual method can additionally run local central-difference gradient steps after each division of the cubes, which improves the estimated optimum within each box and tightens the Lipschitz pruning. Under Assumptions 2.1 and 2.2, these local stages attain rates $\mathcal{O}(M^{-1/2+\varepsilon})$, $\mathcal{O}(1/M)$, and locally linear (up to $q_{\min}^{-1}$ under random block selection), with a uniformly bounded survivor count and sublinear cumulative suboptimality (Appendix B.1; proofs in Appendices F–J).

The guarantees above assume that pruning and evaluation use the exact averaged objective f. The implemented protocol departs from this in three ways: (i) each client searches its own copy of the block locally, pruning with the same Lipschitz test applied to its own loss f_k and then refining toward a local optimum, so Lipschitz pruning protects minimizers of f_k but not those of f: a cell containing a global minimizer of f can still be discarded by a client whose local objective disagrees; (ii) the server combines the reported candidates by loss-weighted least squares rather than exact aggregation; and (iii) a client-vote accept/reject, rather than exact evaluation, decides whether to keep the combined candidate. The guarantees therefore apply to the conceptual method and its randomized blockwise version, not automatically to the implemented protocol. Lemma K.1 gives the condition under which the accept/reject vote is provably aligned with the averaged objective.

5 Communication Efficiency

Each cycle of Algorithm 2 operates on B_{cap} blocks, and each stage costs at most one message per client or one broadcast, so both the uplink and the downlink cost per cycle are bounded by fixed quantities, *independent of* d. Let K be the client count, d' the block dimension, and $n_q = 8$ the quantized word length for every transmitted number (points and losses, in both directions).

Uplink. In each cycle a client sends three messages. The activity mask (Step 2) has B_{cap} bits, one per selected block; it is the only stage every selected block pays for regardless of outcome. Local search (Step 3) is entirely free: no client–server exchange occurs while narrowing and descending. The transmit-gate message (Step 4) is a report/decline mask over the active blocks followed by a quantized report of $n_q(d'+1)$ bits (point plus loss) for each reported block. The vote mask (Step 6) has one bit per block that received a candidate; a block with zero transmitters is a provable no-op and needs no vote. Hence

$$\mathcal{N}_{\mathrm{up}} \;\le\; KB_{\mathrm{cap}}\big(3 + n_q(d'+1)\big). \tag{11}$$

Downlink. In each cycle the server makes three broadcasts: the active set (B_{cap} bits), the combined candidates ($n_q d'$ bits per block with at least one transmitter), and the accept/reject outcomes (one bit per voted block). In addition, it broadcasts one n_s-bit seed per epoch (Assumption 2.4); no block identities are sent. With $N_{\mathrm{cyc}} = \lfloor B/B_{\mathrm{cap}} \rfloor$ cycles per epoch,

$$\mathcal{N}_{\mathrm{down}} \;\le\; B_{\mathrm{cap}}\big(2 + n_q d'\big) + \frac{n_s}{N_{\mathrm{cyc}}}. \tag{12}$$

Summing the two,

$$\mathcal{N}_B \;\le\; B_{\mathrm{cap}}\Big(K\big(3+n_q(d'+1)\big)+2+n_q d'\Big)+\frac{n_s}{N_{\mathrm{cyc}}}, \tag{13}$$

independent of d. With the default configuration (K=10, B_{cap}=30, d'=2, n_q=8), this gives at most $8{,}100$ uplink and 540 downlink bits per cycle, plus the n_s-bit seed once per epoch, at most 0.5 bits per cycle on our benchmarks. The minimum is 330 bits, when every selected block is screened out at the activity pre-screen. The activity screen and the transmit-gate cut most of the uplink in practice. Only the number of cycles per epoch, $N_{\mathrm{cyc}} \approx d/(d'B_{\mathrm{cap}})$, grows with d. The totals in Section 6 exclude the active-set and outcome bits and the seeds (at most $2B_{\mathrm{cap}} = 60$ bits per cycle).

Message batching. Each stage is one message per client or one broadcast carrying the bits of all B_{cap} blocks, which amortizes per-message framing overhead (headers, checksums, etc.).

Uplink versus downlink. The uplink is usually the binding constraint in federated learning (Konečnỳ et al., 2016), although over the internet a broadcast costs K unicasts; BlackBoxFL's advantage also holds for the uplink alone (Shakespeare/NanoLM: 2.8 Mb, against 7.5 Mb for BiCompFL and 48 Mb for DeComFL and BAFFLE).

Quantization. Transmitted numbers use an 8-bit uniform fixed-point (INT8) quantizer (Zhao et al., 2021); clients form estimates locally at full precision (quantization-step details in Appendix K). Our simulations charge this budget but exchange values at full precision, so the reported accuracies are those of the unquantized protocol.

6 Numerical Simulations

We evaluate *BlackBoxFL* on MNIST (LeCun et al.), CIFAR-10 (Krizhevsky, 2009), and character-level language modeling on tiny-Shakespeare (Karpathy, 2015) against state-of-the-art communication-efficient methods; each dataset is randomly partitioned across clients into disjoint local splits.

Unless noted, we run $K = 10$ clients, $d' = 2$, $B_{\mathrm{cap}} = 30$, 2 dyadic-narrowing levels and up to 10 local gradient-descent steps; the activity and transmit-gate thresholds vary by benchmark (Appendix M.1). The server holds no training data and performs no forward passes. We report medians over seeded runs with 5–95% seed intervals; baselines run $1{,}000$ rounds at their reference configurations and BlackBoxFL $1{,}000$ cycles, except 300 BlackBoxFL cycles on CIFAR-10/LeNet5 and $3{,}000$ rounds for every method on CIFAR-10/TinyCIFARWide, without equalizing query budgets (Section 7); we use 100 seeds per method. We use five benchmarks: NanoLM on tiny-Shakespeare ($120{,}128$ parameters); TinyCNN ($9{,}074$) and LeNet5 (Lecun et al., 1998) ($44{,}426$) on MNIST; and LeNet5 ($62{,}006$) and TinyCIFARWide, a wider GroupNorm CNN ($31{,}802$), on CIFAR-10 (architectures in Tables 6, 4, 5, 7, and 8). TinyCNN is also used for the ablations (Appendix N). Communication and compute counts are hardware-independent.

6.1 Comparison with Zero-Order Baselines

We compare against the three federated zero-order baselines closest to our approach: *FedZO* (Fang et al., 2022), *BAFFLE* (Feng et al., 2024), and *DeComFL* (Li et al., 2025); comparisons with the first-order communication-efficient references signSGD, FetchSGD, and BiCompFL are given in Appendix L.1; FeedSign (Cai et al., 2025), which is designed for fine-tuning pretrained models rather than training from scratch, is reported in Appendix L. Table 1 summarizes all five benchmarks, and Figure 1 shows the full trade-off on two of them.

Among the methods in Table 1, BlackBoxFL communicates the least on every benchmark: 7–$25\times$ less than DeComFL, the most communication-efficient baseline, and two to four orders of magnitude less than FedZO and BAFFLE. It also attains the best final metric among zero-order methods on every benchmark. On Shakespeare/NanoLM its perplexity is 11.3 at 3.1 Mb, against 13.1 for DeComFL at 52.8 Mb, and also below that of the first-order signSGD (11.8 at $5{,}045$ Mb; Appendix L.1). On the four image-classification benchmarks it leads the zero-order baselines by 3.6–19.0 accuracy points; for instance, 53.7% versus 43.1% on CIFAR-10/TinyCIFARWide, at 21.6 versus 158 Mb. Two caveats qualify these results. Zero-order optimization trades communication

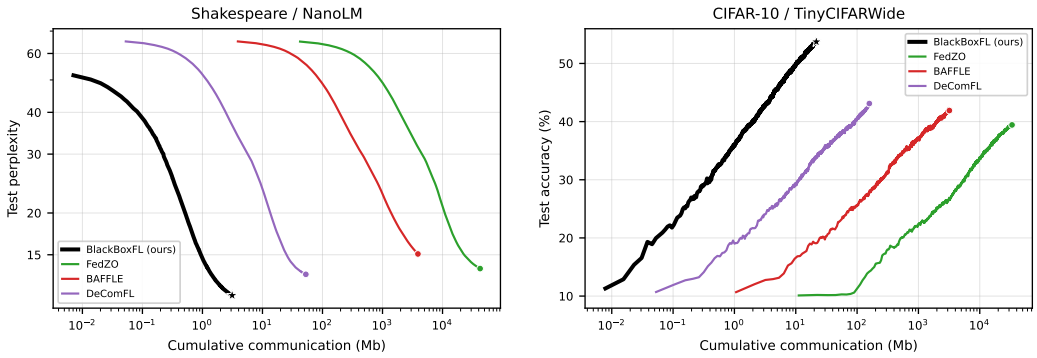


Figure 1: Metric–communication trade-off against the zero-order baselines on Shakespeare/NanoLM (left, perplexity) and CIFAR-10/TinyCIFARWide (right, accuracy): medians over seeds, log-scale axes, markers at final points. BlackBoxFL's curve lies furthest left on both.

Table 1: Zero-order comparison: final test perplexity (Shakespeare, lower is better) or accuracy (%) and total communication (Mb), medians over seeds; best per benchmark in **bold**; full statistics in Table 2.

	Shakespeare/NanoLM		MNIST/TinyCNN		MNIST/LeNet5		CIFAR-10/LeNet5		CIFAR-10/TinyCIFARWide	
Method	Ppl. ↓	Mb	Acc. (%)	Mb	Acc. (%)	Mb	Acc. (%)	Mb	Acc. (%)	Mb
BlackBoxFL (ours)	**11.3**	**3.13**	**96.4**	**4.09**	**95.4**	**6.29**	**39.2**	**2.11**	**53.7**	**21.6**
FedZO	13.6	42,285	90.7	3,194	75.3	15,638	29.6	21,826	39.4	33,583
BAFFLE	15.1	3,892	92.3	338	76.4	1,470	35.3	2,032	41.9	3,197
DeComFL	13.1	52.8	92.4	52.8	75.3	52.8	35.6	52.8	43.1	158

for compute: BlackBoxFL uses 2–22× more forward passes than the zero-order baselines (Table 2). And all methods run at their reference budgets untill converged, rather than matched query budgets (Section 7). Seed intervals and per-benchmark figures are given in Appendix L.

7 Conclusion and Limitations

We presented BlackBoxFL, a communication-efficient federated zero-order protocol whose clients communicate the outcome of a local search over a few low-dimensional parameter blocks, instead of gradients or perturbation-direction estimates, while a client vote on fresh local data replaces server-side validation. Its per-cycle uplink and downlink communication is independent of the model dimension, and on five benchmarks it attains the highest accuracy among the zero-order methods compared at a fraction of their communication. Its design rests on a global-optimization view of federated black-box training: smoothness-matched Lipschitz pruning combined with randomized coordinate coverage makes an exact-loss, fully blockwise method converge to the global optimum under Lipschitz regularity alone, with a bounded number of surviving cells and local rates under quadratic growth.

Our guarantees certify the conceptual partition-based protocol and its randomized-block variant (Proposition 4.4); the weighted-least-squares combination and, above all, the client-vote accept/reject are analyzed only via Section 4's alignment discussion, and vote-based aggregation is known fragile under heterogeneity (Karimireddy et al., 2019; Chen et al., 2020): non-IID clients can unanimously misjudge an update that a genuine held-out check would have caught. This is quantified by our Dirichlet ablation. Two further limitations: (1) Section 6.1's comparisons are communication-centric: BlackBoxFL uses 2–22× more forward passes than the zero-order baselines and 90–490× more than first-order ones, all at reference rather than matched query budgets; and (2) our largest model has 120k parameters; payload is d-independent, coverage is not. The protocol assumes synchronized participation; activity, transmit, and accept/reject bits leak data.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the writing of proofs (checking that our assumptions are valid and that our proofs are correct), to formulate mathematical claims and provide ingredients for proving them (two lemmas bounding when the client accept/reject vote is aligned with the averaged federated objective, together with their proofs, were drafted with AI assistance), to provide feedback on methodology (including identifying and helping resolve an earlier mismatch between the finite-difference estimator used in the analysis and the implementation), to interpret results (computing summary statistics from our run logs), to find recent related work, and to implement methods (refining the Python implementations of the baselines we compare against). We have not used generative AI tools to generate synthetic data sets, to develop the paper's core theoretical framework, to propose or refine hypotheses, to clean or reformat datasets, to support qualitative or thematic data analysis, or to assist with translation. Additionally, we used generative AI tools for tasks with recommended disclosure: drafting and editing parts of this paper, creating and modifying figures, and editing software code. We have reviewed all AI-assisted work: AI-drafted claims, proofs and passages were checked by the authors against the implementation and the underlying run logs, and AI-assisted code was verified and tested. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The protocol is specified completely: Algorithm 1 gives the high-level loop and Algorithm 2 the full pseudocode, including every threshold, the step-size schedule, the quantization width, and the commit rules for both local-search phases. Appendix M.1 lists all hyperparameter values, including the two ($\varepsilon_{\mathrm{act}}$ and $\varepsilon_{\mathrm{tx}}$) that vary by benchmark; Appendix M gives every model architecture layer-by-layer, and Appendix M.2 the hardware and software versions. Reported communication (bits) and compute (forward-pass counts) are logged directly by the protocol and are hardware-independent, so they reproduce exactly rather than approximately; accuracy and perplexity are reported as medians over seeded runs with 5–95% seed intervals. All assumptions used by the theory are stated in Sections 2 and 4, and every proof is given in the appendices.

REFERENCES

Arya Akhavan, Massimiliano Pontil, and Alexandre B. Tsybakov. Distributed zero-order optimization under adversarial noise. In *Advances in Neural Information Processing Systems*, pp. 10209–10220, 2021. arXiv:2102.01121.

Ali Arabzadeh, James A. Grant, and David S. Leslie. Federated $\mathcal{X}$-armed bandit with flexible personalisation. *arXiv preprint arXiv:2409.07251*, 2024.

Burak Bartan and Mert Pilanci. Distributed black-box optimization via error correcting codes. In *57th Annual Allerton Conference on Communication, Control, and Computing*, 2019. arXiv:1907.05984.

Aharon Ben-Tal, Elad Hazan, Tomer Koren, and Shie Mannor. Oracle-based robust optimization via online learning. *Operations Research*, 63(3):628–638, 2015.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018. arXiv:1802.04434.

Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. $\mathcal{X}$-armed bandits. *Journal of Machine Learning Research*, 12:1655–1695, 2011.

HanQin Cai, Yuchen Lou, Daniel McKenzie, and Wotao Yin. A zeroth-order block coordinate descent algorithm for huge-scale black-box optimization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1193–1203, 2021.

Zhijie Cai, Haolong Chen, and Guangxu Zhu. FeedSign: Robust full-parameter federated fine-tuning of large models with extremely low communication overhead of one bit, 2025.

Hongxu Chen, Jinchi Chen, and Ke Wei. A zeroth-order variance-reduced method for decentralized stochastic non-convex optimization. *Optimization*, 2023a. arXiv:2310.18883.

Jun Chen, Hong Chen, Bin Gu, and Hao Deng. Fine-grained theoretical analysis of federated zeroth-order optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023b.

Xiangyi Chen, Tiancong Chen, Haoran Sun, Zhiwei Steven Wu, and Mingyi Hong. Distributed training with heterogeneous data: Bridging median- and mean-based algorithms. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Fedavg with fine tuning: Local updates lead to representation learning. *Advances in Neural Information Processing Systems*, 35:10572–10586, 2022.

Ron Dorfman, Shay Vargaftik, Yaniv Ben-Itzhak, and Kfir Yehuda Levy. Docofl: Downlink compression for cross-device federated learning. In *International Conference on Machine Learning*, pp. 8356–8388. PMLR, 2023.

Maximilian Egger, Rawad Bitar, Antonia Wachter-Zeh, Nir Weinberger, and Deniz Gündüz. Bicompfl: Stochastic federated learning with bi-directional compression. *arXiv preprint arXiv:2502.00206*, 2025.

Wenzhi Fang, Ziyi Yu, Yuning Jiang, Yuanming Shi, Colin N. Jones, and Yong Zhou. Communication-efficient stochastic zeroth-order optimization for federated learning. *IEEE Transactions on Signal Processing*, 70:5058–5073, 2022. arXiv:2201.09531.

Haozhe Feng, Tianyu Pang, Chao Du, Wei Chen, Shuicheng Yan, and Min Lin. BAFFLE: A baseline of backpropagation-free federated learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. arXiv:2301.12195.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Kaja Gruntkowska, Alexander Tyurin, and Peter Richtárik. Ef21-p and friends: Improved theoretical communication complexity for distributed optimization with bidirectional compression. In *International conference on machine learning*, pp. 11761–11807. PMLR, 2023.

Xiaoyu He, Zibin Zheng, Chuan Chen, Yuren Zhou, Chuan Luo, and Qingwei Lin. Distributed evolution strategies for black-box stochastic optimization. *arXiv preprint arXiv:2204.04450*, 2022.

Berivan Isik, Francesco Pase, Deniz Gunduz, Tsachy Weissman, and Michele Zorzi. Sparse random networks for communication-efficient federated learning. *arXiv preprint arXiv:2209.15328*, 2022.

Berivan Isik, Francesco Pase, Deniz Gunduz, Sanmi Koyejo, Tsachy Weissman, and Michele Zorzi. Adaptive compression in federated learning via side information. In *International Conference on Artificial Intelligence and Statistics*, pp. 487–495. PMLR, 2024.

Shinji Ito, Daisuke Hatano, Hanna Sumita, Kei Takemura, Takuro Fukunaga, Naonori Kakimura, and Ken-Ichi Kawarabayashi. Oracle-efficient algorithms for online linear optimization with bandit feedback. *Advances in Neural Information Processing Systems*, 32, 2019.

Donald R. Jones, Cary D. Perttunen, and Bruce E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian U. Stich, and Martin Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. In *International Conference on Machine Learning*, 2019.

Andrej Karpathy. char-rnn: tiny-shakespeare data. `https://raw.githubusercontent.com/karpathy/char-rnn/master/data/tinyshakespeare/input.txt`, 2015. URL `https://github.com/karpathy/char-rnn`.

Jack Kiefer and Jacob Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The mnist database of handwritten digits. `http://yann.lecun.com/exdb/mnist`. URL `http://yann.lecun.com/exdb/mnist`.

Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

Wenjie Li, Qifan Song, and Jean Honorio. Personalized federated $\mathcal{X}$-armed bandit. *arXiv preprint arXiv:2310.16323*, 2023.

Wenjie Li, Qifan Song, Jean Honorio, and Guang Lin. Federated $\mathcal{X}$-armed bandit. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 13628–13636, 2024.

Zhe Li, Bicheng Ying, Zidong Liu, Chaosheng Dong, and Haibo Yang. Achieving dimension-free communication in federated learning via zeroth-order optimization. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2405.15861.

Zhe Li, Bicheng Ying, Zidong Liu, Chaosheng Dong, and Haibo Yang. Converge faster, talk less: Hessian-informed federated zeroth-order optimization. In *International Conference on Learning Representations*, 2026.

Cédric Malherbe and Nicolas Vayatis. Global optimization of lipschitz functions. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Elissa Mhanna and Mohamad Assaad. Single point-based distributed zeroth-order optimization with a non-convex stochastic objective function. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Rémi Munos. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011.

Angelia Nedić and Alex Olshevsky. Distributed optimization over time-varying directed graphs. In *52nd IEEE Conference on Decision and Control (CDC)*, pp. 6855–6860, 2013. arXiv:1303.2289.

Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Chanho Park, H. Vincent Poor, and Namyoon Lee. SignSGD with federated voting, 2024.

Constantin Philippenko and Aymeric Dieuleveut. Bidirectional compression in heterogeneous settings for distributed or federated learning with partial participation: tight convergence guarantees. *arXiv preprint arXiv:2006.14591*, 2020.

Zhen Qin, Daoyuan Chen, Bingchen Qian, Bolin Ding, Yaliang Li, and Shuiguang Deng. Federated full-parameter tuning of billion-sized language models with communication cost under 18 kilobytes. In *International Conference on Machine Learning*, 2024. arXiv:2312.06353.

Zhaonan Qu, Kaixiang Lin, Jayant Kalagnanam, Zhaojian Li, Jiayu Zhou, and Zhengyuan Zhou. Federated learning's blessing: Fedavg has linear speedup. In *ICLR 2021-Workshop on Distributed and Private Machine Learning (DPML)*, 2021.

Ayush Rai and Shaoshuai Mou. Distributed optimization via kernelized multi-armed bandits. *arXiv preprint arXiv:2312.04719*, 2023.

Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11893–11902, 2020.

Yide Ran, Wentao Guo, Jingwei Sun, Yanzhou Pan, Xiaodong Yu, Hao Wang, Jianwen Xie, Yiran Chen, Denghui Zhang, and Zhaozhuo Xu. Mitigating non-IID drift in zeroth-order federated LLM fine-tuning with transferable sparsity. *arXiv preprint arXiv:2506.03337*, 2025.

Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1):1–38, 2014.

Herbert Robbins and David Siegmund. A convergence theorem for non negative almost supermartingales and some applications. In Jagdish S. Rustagi (ed.), *Optimizing Methods in Statistics*, pp. 233–257. Academic Press, 1971.

Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. FetchSGD: Communication-efficient federated learning with sketching. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Yao Shu, Xiaoqiang Lin, Zhongxiang Dai, and Bryan Kian Hsiang Low. Federated zeroth-order optimization using trajectory-informed surrogate gradients. *arXiv preprint arXiv:2308.04077*, 2023.

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. *Advances in neural information processing systems*, 31, 2018.

Hanlin Tang, Xiangru Lian, Chen Yu, Tong Zhang, and Ji Liu. Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. In *International Conference on Machine Learning*, pp. 6155–6165. PMLR, 2019a.

Yujie Tang, Junshan Zhang, and Na Li. Distributed zero-order algorithms for nonconvex multi-agent optimization. *arXiv preprint arXiv:1908.11444*, 2019b.

John Tsitsiklis, Dimitri Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9): 803–812, 1986.

Stephen J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

Cong Xie, Shuai Zheng, Sanmi Koyejo, Indranil Gupta, Mu Li, and Haibin Lin. Cser: Communication-efficient sgd with error reset. *Advances in Neural Information Processing Systems*, 33:12593–12603, 2020.

Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021.

Kang Zhao, Sida Huang, Pan Pan, Yinghan Li, Yingya Zhang, Zhenyu Gu, and Yinghui Xu. Distribution adaptive int8 quantization for training cnns. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 3483–3491, 2021.

Shuai Zheng, Ziyue Huang, and James Kwok. Communication-efficient distributed blockwise momentum sgd with error-feedback. *Advances in Neural Information Processing Systems*, 32, 2019.

A Extended Related Work

This appendix expands the condensed related-work discussion of Section 1.

Gradient and model compression. Seminal approaches include Mem-SGD (Stich et al., 2018), which employs sparsification with memory for bidirectional communication; DoubleSqueeze (Tang et al., 2019a), which leverages double-pass error-compensated compression; and block-wise SignSGD with momentum (Zheng et al., 2019), building on the original signSGD scheme (Bernstein et al., 2018), which compresses each gradient coordinate to a single sign bit and aggregates by majority vote; signSGD-FV (Park et al., 2024) replaces the plain majority by a weighted vote with per-worker weights learned online, addressing heterogeneous minibatch sizes. FetchSGD (Rothchild et al., 2020) instead sketches the accumulated gradient into a fixed-size Count Sketch, decoupling per-round communication cost from the number of model parameters. Philippenko & Dieuleveut established theoretical foundations for bidirectional compression with Artemis (Philippenko & Dieuleveut, 2020), demonstrating tight convergence guarantees. More recent advances include CSER (Xie et al., 2020), DoCoFL (Dorfman et al., 2023), and EF21-P (Gruntkowska et al., 2023).

Network sparsification. Building on the lottery ticket hypothesis (Frankle & Carbin, 2018), which demonstrated that sparse subnetworks within randomly initialized networks can achieve competitive performance, Ramanujan et al. established that randomly weighted networks inherently contain suitable sparse subnetworks (Ramanujan et al., 2020). Isik et al. extended these insights to federated settings through FedPM (Isik et al., 2022), which collaboratively trains probabilistic masks to identify and transmit only essential parameters. Recent work has further leveraged importance sampling to improve communication efficiency: KLMS (Isik et al., 2024) applies importance sampling to stochastic compressors and Bayesian FL settings, while BiCompFL (Egger et al., 2025) extends this to joint uplink and downlink compression, achieving state-of-the-art communication efficiency. BiCompFL is *not* a zero-order method: it relies on full gradient computation at each client to train its mask logits, and we use it in Appendix L.1 as a first-order reference point against which to gauge how close a purely zero-order method can get.

Federated zero-order optimization. Fang et al. proposed FedZO (Fang et al., 2022), a communication-efficient stochastic ZO method for FL with nonconvex convergence guarantees, partial device participation, and an over-the-air computation variant. A subsequent NeurIPS 2023 work (Chen et al., 2023b) provided a finer-grained theoretical analysis of federated ZO. BAFFLE (Feng et al., 2024) removes backpropagation entirely: clients upload only scalar loss differences from forward-pass perturbations, letting the server reconstruct a ZO gradient estimate without ever seeing raw gradients. DeComFL (Li et al., 2025) pushes this further by regenerating perturbation directions from a shared random seed, so only a handful of scalars are exchanged in *both* directions, making communication cost independent of the model dimension. BlackBoxFL also relies on a shared seed (Assumption 2.4), but only to select the blocks each cycle, not to generate perturbation directions. FedKSeed (Qin et al., 2024) applies the same seed mechanism to federated full-parameter fine-tuning of billion-parameter language models with kilobyte-scale total communication, and HiSo (Li et al., 2026) accelerates scalar-only federated ZO optimization with Hessian-informed sampling. FeedSign (Cai et al., 2025) pushes seed-based communication to its extreme: each aggregation step costs exactly one bit in each direction, encoding the sign of a scalar ZO estimate along a shared-seed direction. BlackBoxFL differs in mechanism: besides one-bit control decisions (a per-client transmit/decline decision and an accept/reject vote), its clients report the outcome of a local search over a low-dimensional *region*, driving domain narrowing rather than a step along a pseudo-random direction; it communicates more per round than FeedSign but reaches substantially higher accuracy when training from scratch (Appendix L). FZooS (Shu et al., 2023) further improves query and communication efficiency in heterogeneous FL through trajectory-informed surrogate gradients. In the sparse regime, Meerkat (Ran et al., 2025) applies sparse ZO optimization to federated LLM fine-tuning, mitigating non-IID client drift via transferable sparsity.

Distributed black-box optimization. Distributed optimization has become essential for large-scale problems where data and computation are dispersed across multiple agents. In the black-box (zero-order) setting, agents minimize a global objective using only local function evaluations, particularly relevant when gradients are unavailable, costly, or restricted by privacy constraints (Ito et al., 2019; Ben-Tal et al., 2015). Recent advances have achieved optimal convergence rates in nonconvex scenarios (He et al., 2022; Akhavan et al., 2021), proposed error-correcting frameworks for

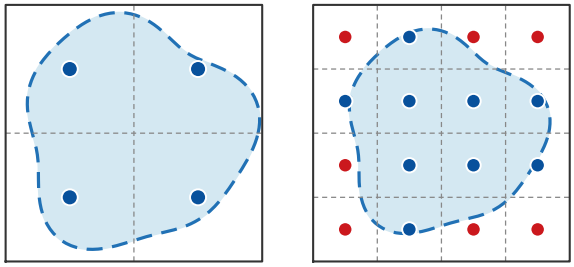


Figure 2: Space subdivision of the *conceptual* full-dimensional method described in this appendix for $d = 2$: left shows stage $m = 2$ with $2^d = 4$ sub-cubes, and right shows stage $m = 3$ with $2^{2d} = 16$ sub-cubes. Under the box-domain formulation, all cell centers are feasible; the dotted contour is illustrative rather than a domain boundary. The implemented protocol does not construct this full partition, but performs per-client local narrowing on d'-dimensional blocks (Section 3).

resilience (Bartan & Pilanci, 2019), and developed multi-agent methods with provable convergence guarantees (Tang et al., 2019b). Kernel-based zero-order methods have been extended to distributed networks (Rai & Mou, 2023), and ZO-BCD (Cai et al., 2021) scales zeroth-order optimization to huge-scale black-box problems by randomized block coordinate descent in the centralized setting, while consensus-based strategies extend to zero-order settings, ensuring privacy preservation and communication efficiency (Tsitsiklis et al., 1986; Nedić & Olshevsky, 2013; Chen et al., 2023a; Mhanna & Assaad, 2023).

B THE CONCEPTUAL FULL-DIMENSIONAL METHOD

The analysis of Section 4 concerns the conceptual method described below on the normalized box $\mathcal{C} = [0, 1]^d$ introduced in Section 2. A grid-refinement scheme subdivides the search space by a factor of two along each axis at every stage; after m stages there are $(2^d)^{m-1}$ sub-cubes with common side length $2^{-(m-1)}$ and diameter $\delta_m = \sqrt{d}\, 2^{-(m-1)} \to 0$ (Figure 2). Sampling every cell center and applying Proposition 4.2 guarantees convergence, but at exponential sampling cost; the conceptual method therefore prunes and refines optimistically.

Time is divided into segments, starting from the candidate list $l_c(1) = \{\mathcal{C}\}$ with the incumbent $\mathbf{x}_{best}^1$ at the center of $\mathcal{C}$; segment n performs one global division-and-sampling stage over the candidate list $l_c(n)$, followed by $k(n)$ local projected zero-order gradient stages (Definition B.2) started from the incumbent best point, where $k(n)$ is a nondecreasing schedule.

Lemma B.1 (Lipschitz bound)**.** *Let f be L_0-Lipschitz. For any sub-cube with center $\mathbf{x}_c$ and diameter δ_n, all $\mathbf{x}$ in the sub-cube satisfy $|f(\mathbf{x}) - f(\mathbf{x}_c)| \le L_0\delta_n/2$. Hence if $f(\mathbf{x}_{best}^n) < f(\mathbf{x}_c) - L_0\delta_n/2$, the sub-cube cannot contain a global minimizer.*

The algorithm prunes with the tighter of the two valid lower bounds: the Lipschitz bound of Lemma B.1 and the smoothness-matched bound of Lemma 4.1, including the margin $\varepsilon_{\text{prune}}$ of Lemma 4.1 when losses are quantized. Sub-cubes surviving the test are collected into $l_c(n)$, each is subdivided into 2^d children, the children's centers (and their finite-difference companions) are evaluated, and the incumbent $\mathbf{x}_{best}^{n+1}$ is updated. In the federated setting, the server broadcasts the children to be sampled, each client evaluates its local loss at their centers and finite-difference companions, and the server averages the returned losses into exact values of f, which the pruning test and the incumbent use.

B.1 LOCAL REFINEMENT RATES AND CUMULATIVE COST

Cumulative suboptimality. For any queried point $\mathbf{x} \in \mathcal{C}$, let $G(\mathbf{x}) := f(\mathbf{x}) - f^* \ge 0$ denote the optimality gap. If $\mathbf{x}^{[j]}$ is the point used in the jth black-box evaluation, the cumulative suboptimality

after N_{ev} evaluations is

$$\mathcal{R}(N_{\mathrm{ev}}) := \sum_{j=1}^{N_{\mathrm{ev}}} G(\mathbf{x}^{[j]}), \tag{14}$$

where every distinct oracle query is counted.

To accelerate the convergence, we propose to relax the algorithm and allow local finite difference gradient search after each division of the cubes. This improves the estimated optimum within each box, and improves the Lipschitz pruning. The quantitative rates require Assumptions 2.1 and 2.2.

Definition B.2 (Projected central-difference refinement)**.** *On a convex retained box C', the local stage uses $\mathbf{x}_{m+1} = \Pi_{C'}(\mathbf{x}_m - \eta_m \widetilde{\nabla} f(\mathbf{x}_m; h_m))$ with $h_m = \mathcal{O}(\eta_m)$, whenever the symmetric probes $\mathbf{x}_m \pm h_m \mathbf{e}_i$ lie in the global feasible domain $\mathcal{C}$ for all coordinates used by the update. The localized rate results are applied after the retained region lies in the interior neighborhood of an interior minimizer, where such a positive symmetric radius exists.*

Theorem B.3 (Localized best-iterate rates)**.** *If $C' \subseteq \mathbb{B}(\mathbf{x}^*, \rho)$ contains a global minimizer and $h_m \le \operatorname{dist}(C', \partial\mathcal{C})$, then Definition B.2 achieves $\mathcal{O}(M^{-1/2+\varepsilon})$ for $\eta_m = m^{-1/2-\varepsilon}$, $\varepsilon \in (0, \frac{1}{2})$, and $\mathcal{O}(1/M)$ for the μ-adapted schedule $\eta_m = 4/(\mu(m+1))$.*

Theorem B.4 (Local linear rate)**.** *Under the same localized assumptions, a constant-step projected full-gradient iteration is locally linear; central differences with feasible symmetric probes retain linear convergence when their step decreases geometrically.*

Theorem B.5 (Randomized blockwise local refinement)**.** *If every block has conditional inclusion probability at least $q_{\min} > 0$, then the localized randomized block finite-difference stage converges almost surely and attains expected best-iterate rates $\mathcal{O}(q_{\min}^{-1} M^{-1/2+\varepsilon})$ and $\mathcal{O}((q_{\min} M)^{-1})$ under the two schedules above.*

Theorem B.6 (Survivor count and cumulative cost)**.** *Under Assumptions 2.1 and 2.2, exact evaluations, Lipschitz pruning, and $h_n \le \delta_n/\sqrt{d}$, the number of surviving cells is uniformly bounded after a finite stage, although the bound is exponential in d.*

Theorem B.7 (Bounded cumulative suboptimality)**.** *Under the same conditions, the cumulative suboptimality cost is sublinear for nondecreasing local-stage budgets $k(n)$ and is finite whenever $\sum_n k(n)\delta_n^2 < \infty$.*

C DETAILED BLACKBOXFL PROTOCOL

Algorithm 2 BlackBoxFL: Communication-Efficient Block Protocol

Input: Clients K; block dimension d'; block budget B_{cap}; box radius ρ_0; dyadic levels Λ; max. gradient steps M_{gd}; gradient step schedule $\eta_0, c_\eta, \eta_{\min}$; gradient-norm stopping threshold $\varepsilon_{\mathrm{conv}}$ and minimum per-step improvement $\varepsilon_{\mathrm{gstep}}$; finite-difference step h with box-relative scale c_h and floor $h_{\min}$; activity threshold $\varepsilon_{\mathrm{act}}$ and majority ν_{act}; transmit-gate threshold $\varepsilon_{\mathrm{tx}}$; accept/reject majority ν_{acc}; quantization width n_q.
Init: Set $\eta \leftarrow \eta_0$; initialize each client's last reported loss λ_k to its local loss at the initial model (uploaded once, n_q bits per client).
for epoch $t = 1, 2, \ldots$ **do**
  Server: Broadcast a fresh n_s-bit seed. **All clients:** from it, reshuffle the coordinates within each layer, partition them into layer-aligned blocks $\{I_b\}_{b=1}^{B}$, and draw a random permutation of all blocks.
  while at least B_{cap} blocks of the permutation are undrawn **do**
    — Step 1: Block selection (random permutation, blind) —
    All clients: Take the next B_{cap} blocks of the shared permutation (nothing is transmitted). No history, score, or ranking of any kind is used.
    — Step 2: Activity screening —
    for client $k = 1, \ldots, K$ **do**
      For each selected block b, on a local minibatch, compute the central-difference block estimate $\widehat{z}_{kb} \in \mathbb{R}^{d'}$, with $(\widehat{z}_{kb})_i = \big[f_k(\mathbf{x} + h\mathbf{e}_i) - f_k(\mathbf{x} - h\mathbf{e}_i)\big]/2h$ for $i \in I_b$ ($2d'$ forward passes; see the remark on estimator choice in Appendix K); set one bit indicating $\|\widehat{z}_{kb}\| \geq \varepsilon_{\mathrm{act}}$. Send the B_{cap} activity bits in one message.
    end for
    Server: Mark a block active if at least $\nu_{\mathrm{act}} K$ clients report activity; skip the others for this cycle (no further steps, no transmission, no vote). Broadcast the set of active blocks.
    — Step 3: Local search (per client, free) —
    for client $k = 1, \ldots, K$ **in parallel, no communication do**
      for each active block b **do**
        Record the baseline loss at b's current value $\mathbf{x}_{I_b}$ on a local minibatch.
        Dyadic narrowing: starting from the block's box Q_b, for $\ell = 1, \ldots, \Lambda$ halve the box toward whichever candidate midpoint has lower local loss; **commit the halving only if it improves on the current point**, otherwise keep the box and point unchanged (first fix: no unconditional commit).
        Gradient descent: from the dyadic result, for up to M_{gd} steps compute the central-difference block gradient with $h = \mathrm{clamp}(c_h \cdot \text{box width},\ h_{\min},\ h)$, take a step of size η clamped to Q_b, and **commit the step only if it strictly improves the local loss**, otherwise revert to the last good point and stop early (second fix: no unconditional commit). Stop early also once the estimated gradient norm falls below $\varepsilon_{\mathrm{conv}}$, or the per-step relative improvement falls below $\varepsilon_{\mathrm{gstep}}$.
      end for
    end for
    — Step 4: Transmit-gate (per client) —
    for client $k = 1, \ldots, K$ **do**
      For each active block b: set its bit in a report/decline mask to "report" if baseline loss $-$ final local loss $\geq \varepsilon_{\mathrm{tx}}$, and then include the quantized (n_q bits/coordinate) local point and loss and update λ_k to this loss; otherwise set it to "decline". Send the mask and the reports in one message.
    end for
    Active blocks where no client transmitted are provable no-ops: leave them unchanged and skip Steps 5–6 for them.
    — Step 5: Combination —
    Server: For each remaining active block b, form $\mathbf{x}_b^+ = \big(\sum_k \mathbf{x}_k/\lambda_k\big)/\big(\sum_k 1/\lambda_k\big)$ over all K clients, where a transmitting client contributes its reported point and loss and a declining client contributes the current value $\mathbf{x}_{I_b}$ with its last reported loss λ_k. Broadcast these candidates in one message.
    — Step 6: Accept/reject (client vote) —
    for client $k = 1, \ldots, K$ **do**
      For each block that received a candidate, draw a fresh local minibatch; evaluate $\mathbf{x}_b^+$ and $\mathbf{x}_{I_b}$ on it; set one bit indicating whether $\mathbf{x}_b^+$ improved. Send all votes in one message.
    end for
    Server: For each block that received a candidate, commit $\mathbf{x}_{I_b} \leftarrow \mathbf{x}_b^+$ if at least $\nu_{\mathrm{acc}} K$ clients vote improved, otherwise revert ($\mathbf{x}_{I_b}$ unchanged). Broadcast the outcome for each block.
    $\eta \leftarrow \max(c_\eta \eta, \eta_{\min})$ (once per cycle).
  end while
  The remaining $B \bmod B_{\mathrm{cap}}$ blocks of the permutation are skipped for this epoch.
end for

Weighted least-squares combination. For a block b with at least one report, the server forms

$$\mathbf{x}_b^+ = \arg\min_{\mathbf{v}} \sum_{k=1}^{K} w_k \|\mathbf{v} - \mathbf{x}_k\|_2^2 = \frac{\sum_k w_k \mathbf{x}_k}{\sum_k w_k}, \qquad w_k = \frac{1}{\max(\lambda_k, \epsilon)}, \quad \epsilon = 10^{-6},$$

over all K clients: a transmitting client contributes its locally optimized point $\mathbf{x}_k$ and the local loss λ_k it reports there; a declining client contributes the current value $\mathbf{x}_{I_b}$ and its most recently reported loss (initialized to its local loss at the initial model). No clipping or robust estimator is applied. Outliers are limited by the protocol instead: every $\mathbf{x}_k$ lies in the block's search box Q_b, so the candidate, a convex combination, does too, and a single client can move it by at most ρ_0 per coordinate; only clients whose local loss improved by at least $\varepsilon_{\mathrm{tx}}$ transmit; decliners anchor the candidate at the current value; and the client vote rejects any candidate that a majority finds worse on fresh local data. A misreported loss can still skew the weights; robust alternatives (e.g., a weighted median) are left for future work.

D Lemma 2.3 Proof

Proof. If $\mathbf{x}^*, \mathbf{y}^* \in X^*$ with $\mathbf{y}^* \in \mathbb{B}(\mathbf{x}^*, \rho)$, quadratic growth gives $0 = f(\mathbf{y}^*) - f(\mathbf{x}^*) \geq \frac{\mu}{2}\|\mathbf{y}^* - \mathbf{x}^*\|^2$, so $\mathbf{y}^* = \mathbf{x}^*$. Partition $[0,1]^d$ into $(\lfloor \sqrt{d}/\rho \rfloor + 1)^d$ congruent cubes of side $s < \rho/\sqrt{d}$, hence of diameter less than ρ; each contains at most one global minimizer. □

E Proposition 4.2 Proof

Proof. Fix $\mathbf{x}^* \in X^*$ and let $\mathbf{y}_m \in G_m$ with $\|\mathbf{y}_m - \mathbf{x}^*\| \leq c_m$, where $c_m \to 0$ is the covering radius of G_m. By definition of $\mathbf{x}_m$ and continuity of f on the compact $\mathcal{C}$, $f(\mathbf{x}^*) \leq f(\mathbf{x}_m) \leq f(\mathbf{y}_m) \to f(\mathbf{x}^*)$, so $f(\mathbf{x}_m) \to f(\mathbf{x}^*)$. If $\mathrm{dist}(\mathbf{x}_m, X^*) \not\to 0$, compactness yields a subsequence $\mathbf{x}_{m_j} \to \mathbf{z}$ with $\mathrm{dist}(\mathbf{x}_{m_j}, X^*) \geq \varepsilon$ for some $\varepsilon > 0$; since the distance is continuous, $\mathrm{dist}(\mathbf{z}, X^*) \geq \varepsilon$, so $\mathbf{z} \notin X^*$; continuity gives $f(\mathbf{z}) = f(\mathbf{x}^*)$, contradicting $\mathbf{z} \notin X^*$. When X^* is a singleton, every convergent subsequence has the same limit, so $\mathbf{x}_m \to \mathbf{x}^*$. □

F Theorems B.3 and B.4 Proofs

Proof of Theorem B.3. **Part (i).** On $C' \subseteq \mathbb{B}(\mathbf{x}^*, \rho)$, f is convex (Assumption 2.2) and L_0-Lipschitz, so $\|\nabla f\|_2 \leq L_0$. Write $\tilde{g}_m = \nabla f(\mathbf{x}_s[m]) + \mathbf{b}_m$, where the finite-difference bias satisfies $\|\mathbf{b}_m\|_2 \leq \frac{\sqrt{d}}{2} L_1 h_m$. Since $\mathbf{x}^* \in C'$ and the projection $\Pi_{C'}$ onto the convex set C' is nonexpansive,

$$\begin{aligned} \|\mathbf{x}_s[m+1] - \mathbf{x}^*\|_2^2 &\leq \|\mathbf{x}_s[m] - \eta(m)\tilde{g}_m - \mathbf{x}^*\|_2^2 \\ &= \|\mathbf{x}_s[m] - \mathbf{x}^*\|_2^2 \\ &\quad - 2\eta(m)\langle \tilde{g}_m,\, \mathbf{x}_s[m] - \mathbf{x}^* \rangle \\ &\quad + \eta^2(m)\|\tilde{g}_m\|_2^2. \end{aligned} \tag{15}$$

By convexity, $\langle \nabla f(\mathbf{x}_s[m]),\, \mathbf{x}_s[m] - \mathbf{x}^* \rangle \geq f(\mathbf{x}_s[m]) - f(\mathbf{x}^*)$; with $D = \mathrm{diam}(\mathcal{C})$, Cauchy–Schwarz gives $\langle \tilde{g}_m,\, \mathbf{x}_s[m] - \mathbf{x}^* \rangle \geq f(\mathbf{x}_s[m]) - f(\mathbf{x}^*) - \|\mathbf{b}_m\|_2 D$. Using $\|\tilde{g}_m\|_2 \leq L_0 + \frac{\sqrt{d}}{2} L_1 h_m =: G_{\max}$,

$$\begin{aligned} \|\mathbf{x}_s[m+1] - \mathbf{x}^*\|_2^2 \leq \|\mathbf{x}_s[m] - \mathbf{x}^*\|_2^2 & \\ - 2\eta(m)\big(f(\mathbf{x}_s[m]) - f(\mathbf{x}^*)\big) & \\ + \eta^2(m) G_{\max}^2 + 2D\,\eta(m)\|\mathbf{b}_m\|_2. & \end{aligned} \tag{16}$$

Summing (16) over $m = 1, \ldots, M$ and using $\|\mathbf{x}_s[M+1] - \mathbf{x}^*\|_2^2 \geq 0$,

$$\begin{aligned} &\min_{1 \leq m \leq M} \big(f(\mathbf{x}_s[m]) - f(\mathbf{x}^*)\big) \;\leq \\ &\frac{\|\mathbf{x}_s[1] - \mathbf{x}^*\|_2^2 + G_{\max}^2 \sum_{m=1}^{M} \eta^2(m)}{2\sum_{m=1}^{M} \eta(m)} + \frac{\sqrt{d} L_1 D \sum_{m=1}^{M} \eta(m) h_m}{2\sum_{m=1}^{M} \eta(m)}. \end{aligned} \tag{17}$$

With $\eta(m) = m^{-1/2-\varepsilon}$, $\varepsilon \in (0, \frac{1}{2})$, and $h_m = \mathcal{O}(\eta(m))$, both $\sum_m \eta^2(m)$ and $\sum_m \eta(m)h_m$ are convergent series whose value is $\Theta(1/\varepsilon)$, while $\sum_{m\le M} \eta(m) = \Theta\big(M^{1/2-\varepsilon}\big)$ (at the excluded endpoint $\varepsilon = \frac{1}{2}$ this sum grows only as $\Theta(\log M)$, which is why the interval is open). Hence

$$\min_{1\le m\le M} \big(f(\mathbf{x}_s[m]) - f(\mathbf{x}^*)\big) = \mathcal{O}\big(M^{-1/2+\varepsilon}\big), \tag{18}$$

with hidden constant $\Theta(1/\varepsilon)$, as claimed. Since the incumbent $\mathbf{x}_{best}$ tracks the best sampled point, the bound applies to the algorithm's output.

Part (ii). Write $\Delta_m := f(\mathbf{x}_s[m]) - f(\mathbf{x}^*)$, $r_m := \|\mathbf{x}_s[m] - \mathbf{x}^*\|_2$, and $\beta_m := \frac{\sqrt{d}}{2} L_1 h_m \ge \|\mathbf{b}_m\|_2$. All iterates remain in $C' \subseteq \mathbb{B}(\mathbf{x}^*, \rho)$ by projection, so the quadratic growth of Assumption 2.2 applies: $\Delta_m \ge \frac{\mu}{2} r_m^2$. Splitting the convexity bound in half and inserting growth,

$$\langle \nabla f(\mathbf{x}_s[m]),\, \mathbf{x}_s[m] - \mathbf{x}^* \rangle \;\ge\; \Delta_m \;\ge\; \tfrac{1}{2}\Delta_m + \tfrac{\mu}{4} r_m^2,$$

and $\langle \mathbf{b}_m, \mathbf{x}_s[m] - \mathbf{x}^* \rangle \ge -\beta_m D$, so the descent recursion of Part (i) sharpens to

$$\begin{aligned} r_{m+1}^2 \;&\le \big(1 - \tfrac{\mu}{2}\eta(m)\big) r_m^2 \;-\; \eta(m)\Delta_m \\ &\quad +\; 2D\,\eta(m)\beta_m \;+\; \eta^2(m) G_{\max}^2. \end{aligned}$$

Insert $\eta(m) = 4/(\mu(m+1))$, so that $1 - \frac{\mu}{2}\eta(m) = \frac{m-1}{m+1}$, and multiply through by $m(m+1)$: using $m(m+1)\eta(m) = 4m/\mu$ and $m(m+1)\eta^2(m) = \frac{16}{\mu^2} \cdot \frac{m}{m+1} \le \frac{16}{\mu^2}$,

$$\begin{aligned} m(m+1)\, r_{m+1}^2 \;&\le m(m-1)\, r_m^2 \;-\; \tfrac{4m}{\mu}\Delta_m \\ &\quad +\; \tfrac{8D}{\mu}\, m\beta_m \;+\; \tfrac{16}{\mu^2} G_{\max}^2. \end{aligned}$$

Summing over $m = 1, \dots, M$ telescopes (the m=1 term vanishes), and $M(M+1)r_{M+1}^2 \ge 0$ gives

$$\sum_{m=1}^{M} m\,\Delta_m \;\le\; 2D \sum_{m=1}^{M} m\beta_m \;+\; \frac{4MG^2}{\mu}.$$

Since $h_m \le c_h/m$, we have $\sum_{m\le M} m\beta_m \le \frac{\sqrt{d}}{2} L_1 c_h M$, while $\sum_{m\le M} m = M(M+1)/2$; hence

$$\begin{aligned} \min_{1\le m\le M} \Delta_m \;&\le \frac{\sum_{m\le M} m\,\Delta_m}{\sum_{m\le M} m} \\ &\le \frac{2\sqrt{d}\, L_1 D\, c_h + 8G_{\max}^2/\mu}{M+1}, \end{aligned}$$

as claimed. A stochastic treatment of minibatch evaluation noise is left to future work. □

Proof of Theorem B.4. Write $r_m := \|\mathbf{x}_s[m] - \mathbf{x}^*\|$ and $\eta := \frac{1}{2L_1}$. Since $\mathbf{x}^*$ is interior (Assumption 2.2), $\nabla f(\mathbf{x}^*) = 0$.

Step 1 (a PL-type inner-product bound). For $\mathbf{x} \in C' \subseteq \mathbb{B}(\mathbf{x}^*, \rho/2)$, smoothness gives $\|\nabla f(\mathbf{x})\| = \|\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}^*)\| \le L_1 r \le L_1\rho/2$, so the descent-lemma point $\mathbf{y} = \mathbf{x} - \nabla f(\mathbf{x})/L_1$ satisfies $\|\mathbf{y} - \mathbf{x}^*\| \le 2r \le \rho$, i.e. $\mathbf{y} \in \mathbb{B}(\mathbf{x}^*, \rho) \subseteq \mathcal{C}$ (Assumption 2.2). Hence $f(\mathbf{x}^*) \le f(\mathbf{y}) \le f(\mathbf{x}) - \frac{1}{2L_1}\|\nabla f(\mathbf{x})\|^2$, giving $f(\mathbf{x}) - f(\mathbf{x}^*) \ge \frac{1}{2L_1}\|\nabla f(\mathbf{x})\|^2$. Combining with convexity and quadratic growth,

$$\begin{aligned} \langle \nabla f(\mathbf{x}), \mathbf{x} - \mathbf{x}^* \rangle \;&\ge\; f(\mathbf{x}) - f(\mathbf{x}^*) \\ &\ge\; \tfrac{1}{2} \cdot \tfrac{\mu}{2} r^2 + \tfrac{1}{2} \cdot \tfrac{1}{2L_1} \|\nabla f(\mathbf{x})\|^2 \\ &=\; \tfrac{\mu}{4} r^2 + \tfrac{1}{4L_1} \|\nabla f(\mathbf{x})\|^2. \end{aligned}$$

Step 2 (contraction). Since $\mathbf{x}^* \in C'$ and $\Pi_{C'}$ is nonexpansive,

$$\begin{aligned} r_{m+1}^2 \;&\le\; \|\mathbf{x}_s[m] - \eta \nabla f(\mathbf{x}_s[m]) - \mathbf{x}^*\|^2 \\ &=\; r_m^2 - 2\eta\, \langle \nabla f, \mathbf{x}_s[m] - \mathbf{x}^* \rangle + \eta^2 \|\nabla f\|^2, \end{aligned}$$

and inserting Step 1 with $\eta = \frac{1}{2L_1}$, the gradient terms cancel exactly ($-2\eta \cdot \frac{1}{4L_1} + \eta^2 = 0$), leaving $r_{m+1}^2 \le \big(1 - \frac{\mu}{4L_1}\big) r_m^2$. Induction gives the distance bound, and L_1-smoothness with $\nabla f(\mathbf{x}^*) = 0$ gives $f(\mathbf{x}_s[m]) - f(\mathbf{x}^*) \le \frac{L_1}{2} r_m^2$, the function bound.

Step 3 (finite-difference variant). With $\tilde{g}_m = \nabla f(\mathbf{x}_s[m]) + \mathbf{b}_m$, $\|\mathbf{b}_m\| \le \frac{\sqrt{d}}{2} L_1 h_m$, nonexpansiveness and the triangle inequality give

$$\begin{aligned} r_{m+1} \;&\le\; \|\mathbf{x}_s[m] - \eta \nabla f - \mathbf{x}^*\| + \eta \|\mathbf{b}_m\| \\ &\le\; \rho_c \, r_m + \tfrac{\sqrt{d}}{4} h_m, \qquad \rho_c := \big(1 - \tfrac{\mu}{4L_1}\big)^{1/2}. \end{aligned}$$

With $h_m = h_1 \gamma^m$ and $\gamma \in (\rho_c, 1)$, unrolling yields

$$r_{m+1} \;\le\; \rho_c^m r_1 + \tfrac{\sqrt{d} h_1}{4} \sum_{j=1}^{m} \rho_c^{m-j} \gamma^j \;\le\; \gamma^m \Big(r_1 + \tfrac{\sqrt{d} h_1 \gamma}{4(\gamma - \rho_c)} \Big),$$

so $r_m = \mathcal{O}(\gamma^m)$ and, by smoothness, $f(\mathbf{x}_s[m]) - f(\mathbf{x}^*) = \mathcal{O}(\gamma^{2m})$. □

G Lemma 4.1 and Theorem B.6 Proofs

Proof of Lemma 4.1. By the descent lemma, for $\mathbf{x}$ in the sub-cube ($\|\mathbf{x} - \mathbf{x}_c\| \le \delta/2$),

$$f(\mathbf{x}) \ge f(\mathbf{x}_c) - \|\nabla f(\mathbf{x}_c)\| \tfrac{\delta}{2} - \tfrac{L_1}{2} \big(\tfrac{\delta}{2}\big)^2.$$

The coordinatewise central-difference estimate satisfies $|\tilde{g}_{c,i} - \nabla_i f(\mathbf{x}_c)| \le L_1 h/2$ by (7); because $\mathbf{x}_c$ is the cell center and $h \le \delta/(2\sqrt{d})$, all symmetric probes remain in the sub-cube, hence $\|\nabla f(\mathbf{x}_c)\| \le \|\tilde{g}_c\| + \frac{\sqrt{d}}{2} L_1 h$, which yields (9). If a sub-cube contains $\mathbf{x}^* \in X^*$, then $\mathrm{LB}(\mathbf{x}_c) \le f(\mathbf{x}^*) \le f(\mathbf{x}_{best}^n)$. Now let losses be observed with error at most $\varepsilon_{\mathrm{eval}}$, and let $\hat{f}$, $\hat{g}_c$, $\widehat{\mathrm{LB}}$ denote the quantities computed from the observed losses. Each central-difference coordinate obeys $|\hat{g}_{c,i} - \tilde{g}_{c,i}| \le \varepsilon_{\mathrm{eval}}/h$, so $\|\hat{g}_c\| \ge \|\tilde{g}_c\| - \sqrt{d}\, \varepsilon_{\mathrm{eval}}/h$, and therefore $\widehat{\mathrm{LB}}(\mathbf{x}_c) \le \mathrm{LB}(\mathbf{x}_c) + \varepsilon_{\mathrm{eval}} + \sqrt{d}\, \varepsilon_{\mathrm{eval}}\, \delta/(2h)$. Combining, $\hat{f}(\mathbf{x}_{best}^n) \ge f(\mathbf{x}_{best}^n) - \varepsilon_{\mathrm{eval}} \ge \mathrm{LB}(\mathbf{x}_c) - \varepsilon_{\mathrm{eval}} \ge \widehat{\mathrm{LB}}(\mathbf{x}_c) - \varepsilon_{\mathrm{eval}}\big(2 + \sqrt{d}\, \delta/(2h)\big) = \widehat{\mathrm{LB}}(\mathbf{x}_c) - \varepsilon_{\mathrm{prune}}$, so the margined test never discards that sub-cube. Note the h^{-1} amplification of evaluation error through the finite differences: with the choice $h_n = \vartheta \delta_n / \sqrt{d}$ of Theorem B.6, the margin is exactly $\varepsilon_{\mathrm{eval}}(2 + d/(2\vartheta))$. □

Proof of Theorem B.6. Step 1 (localization). Let $\mathbf{x}_c$ be a surviving center at segment n and write $h = h_n$. Survival means $\hat{f}(\mathbf{x}_{best}^n) \ge \widehat{\mathrm{LB}}(\mathbf{x}_c)$, i.e.

$$\begin{aligned} f(\mathbf{x}_c) - f(\mathbf{x}^*) \le\; & \big(f(\mathbf{x}_{best}^n) - f(\mathbf{x}^*)\big) \\ & + \Big(\|\tilde{g}_c\| + \tfrac{\sqrt{d}}{2} L_1 h\Big) \tfrac{\delta_n}{2} + \tfrac{L_1}{8} \delta_n^2. \end{aligned} \tag{19}$$

The sub-cube containing some $\mathbf{x}^* \in X^*$ is never pruned (Lemma 4.1), and its center $\mathbf{x}_c^{opt}$ satisfies, by L_1-smoothness and $\nabla f(\mathbf{x}^*) = 0$ ($\mathbf{x}^*$ interior by Assumption 2.2),

$$f(\mathbf{x}_{best}^n) - f(\mathbf{x}^*) \le f(\mathbf{x}_c^{opt}) - f(\mathbf{x}^*) \le \tfrac{L_1}{2} \big(\tfrac{\delta_n}{2}\big)^2 = \tfrac{L_1}{8} \delta_n^2.$$

Since $\|\tilde{g}_c\| \le L_0 + \frac{\sqrt{d}}{2} L_1 h_n$ and $h_n = \vartheta \delta_n / \sqrt{d} \le \delta_n / (2\sqrt{d})$, the right side of (19) tends to 0 uniformly over survivors as $\delta_n \to 0$. By Assumption 2.2, the finiteness of X^* (Lemma 2.3), compactness, and continuity, there exists n_1 such that for all $n \ge n_1$ every surviving center lies within ρ of X^*.

Step 2 (quadratic-growth radius). For $n \ge n_1$, let $r = \mathrm{dist}(\mathbf{x}_c, X^*) \le \rho$. Smoothness with $\nabla f(\mathbf{x}^*) = 0$ gives $\|\nabla f(\mathbf{x}_c)\| \le L_1 r$, hence $\|\tilde{g}_c\| \le L_1 r + \frac{\sqrt{d}}{2} L_1 h_n \le L_1 r + \frac{L_1}{4} \delta_n$ using $h_n = \vartheta \delta_n / \sqrt{d}$ and $\vartheta \le \frac{1}{2}$. Substituting into (19) together with the quadratic growth $f(\mathbf{x}_c) - f(\mathbf{x}^*) \ge \frac{\mu}{2} r^2$ of Assumption 2.2:

$$\tfrac{\mu}{2} r^2 \;\le\; \tfrac{L_1}{8} \delta_n^2 + \big(L_1 r + L_1 \delta_n\big) \tfrac{\delta_n}{2} + \tfrac{L_1}{8} \delta_n^2 \;\le\; \tfrac{L_1}{2}\, r\, \delta_n + \tfrac{3L_1}{4}\, \delta_n^2. \tag{20}$$

Solving the quadratic inequality $\frac{\mu}{2}r^2 - \frac{L_1}{2}\delta_n r - \frac{3L_1}{4}\delta_n^2 \le 0$ for r and using $\mu \le L_1$,

$$r \;\le\; \frac{\frac{L_1}{2}\delta_n + \sqrt{\frac{L_1^2}{4}\delta_n^2 + \frac{3\mu L_1}{2}\delta_n^2}}{\mu} \;\le\; \frac{L_1 + \sqrt{3\mu L_1/2}}{\mu}\,\delta_n \le\; 3\,\frac{L_1}{\mu}\,\delta_n \;=:\; R_n.$$

Step 3 (packing). In the normalized coordinates of Section 2, $\mathcal{C} = [0,1]^d$. Hence all level-n cells share side $s_n = \delta_n/\sqrt{d}$, and surviving sub-cubes are disjoint cells whose centers lie on a rectangular grid of spacing s_n along each axis. The number of such grid points within Euclidean distance R_n of any fixed point is at most $\big(2R_n\sqrt{d}/\delta_n + 1\big)^d = \big(6\sqrt{d}L_1/\mu + 1\big)^d = \kappa^d$. Summing over the $g = |X^*|$ minimizers (Lemma 2.3) gives $|l_c(n)| \le g\,\kappa^d$, and each survivor spawns 2^d children, so $N_c(n) \le g\,(2\kappa)^d$. Take $n_0 = n_1$. □

H THEOREM B.7 PROOF

Proof of Theorem B.7. Fix $n > n_0$ and write $R_n = 3(L_1/\mu)\delta_n$ for the survivor radius of Theorem B.6.

Sampling cost. The centers sampled at segment n are children of the survivors of segment $n-1$, hence lie within $R_{n-1} + \delta_{n-1}$ of X^*; their $2d$ central-difference companions lie within a further $h_n \le \delta_{n-1}/(2\sqrt{d}) \le \delta_{n-1}$. By L_1-smoothness and $\nabla f = 0$ on X^*, each of the $N_c(n)(2d+1) \le g(2\kappa)^d(2d+1)$ evaluations costs at most

$$\tfrac{L_1}{2}\big(R_{n-1} + 2\delta_{n-1}\big)^2 \le \tfrac{L_1}{2}\big(3\tfrac{L_1}{\mu} + 2\big)^2\,\delta_{n-1}^2 \;=:\; c'\,\delta_{n-1}^2,$$

so $\mathcal{R}^s(n) \le c_1\delta_{n-1}^2$ with $c_1 = g(2\kappa)^d(2d+1)\,c'$.

Refinement cost. Local stages are projected onto the active sub-cube, all of whose points lie within $R_n + \delta_n$ of X^*; each of the $k(n)(2d+1)$ evaluations therefore costs at most $c'\delta_n^2$, giving $\mathcal{R}^{GD}(n) \le c_2\,k(n)\,\delta_n^2$ with $c_2 = (2d+1)\,c'$.

(i) Sublinearity. Every evaluation performed in segment $n > n_0$ has instantaneous cost at most $u_n := c'\,\delta_{n-1}^2$, and $u_n \to 0$. Given $\varepsilon > 0$, choose $n_\varepsilon > n_0$ with $u_n < \varepsilon$ for all $n \ge n_\varepsilon$; the segments $1, \dots, n_\varepsilon$ contribute a finite cost $\mathcal{R}(n_\varepsilon)$, so for T rounds $\mathcal{R} \le \mathcal{R}(n_\varepsilon) + \varepsilon T$ and $\limsup_T \mathcal{R}/T \le \varepsilon$. As ε was arbitrary, $\mathcal{R} = o(T)$ for any nondecreasing schedule $k(n)$.

(ii) Boundedness. Summing the per-segment bounds,

$$\mathcal{R} \;\le\; \mathcal{R}(n_0) + \sum_{n>n_0}\big(c_1\delta_{n-1}^2 + c_2\,k(n)\,\delta_n^2\big).$$

With $\delta_n = \Theta(2^{-n})$ the first sum is geometric, and $k(n) = \mathcal{O}(4^n/n^{1+\varepsilon'})$ with $\varepsilon' > 0$ gives $k(n)\delta_n^2 = \mathcal{O}(n^{-(1+\varepsilon')})$, a convergent series; hence $\mathcal{R} < \infty$. (ε' is distinct from the step-size constant ε of Definition B.2.) Both conclusions hold from n_0 onward and rest on Assumption 2.2 and the projected zero-order dynamics; the pre-localization cost $\mathcal{R}(n_0)$ is finite but can be exponential in d. □

I PROPOSITION 4.4 PROOF

Proof. For each coordinate i, the indicators $X_{i,m}$ that i is refined at stage m are independent across m with $\mathbb{P}(X_{i,m} = 1) \ge p$. By Kolmogorov's strong law for independent bounded variables, $\frac{1}{n}\sum_{m\le n}\big(X_{i,m} - \mathbb{E}X_{i,m}\big) \to 0$ almost surely, so $\liminf_n N_i(n)/n \ge p$; in particular $N_i(n) \to \infty$ for every i on an event of probability one. Each refinement of coordinate i halves the side of every retained cell along axis i, so in normalized coordinates that side is $2^{-N_i(n)}$ and $\delta_n^2 \le \sum_{i=1}^d 4^{-N_i(n)} \to 0$ almost surely. By Lipschitz pruning (Lemma 4.1 with exact evaluations), at every segment at least one retained cell intersects X^*; its sampled center lies within δ_n of X^*, so by

continuity the best sampled value converges to $f(\mathbf{x}^*)$ almost surely, and when the global minimizer is unique the best sampled point converges to it (Proposition 4.2). Finally, since $\liminf_n N_i(n)/n \geq p$ almost surely, for any $p' < p$ there is almost surely a finite n' such that $N_i(n) \geq p'n$ for every i and all $n \geq n'$, whence $\delta_n \leq \sqrt{d}\, 2^{-p'n}$ for all $n \geq n'$. □

J Proposition 4.3, Theorem B.5, and Corollary 4.5 Proofs

Proof of Proposition 4.3. Let $A_S : \mathbb{R}^{d_S} \to \mathbb{R}^d$, $A_S(\mathbf{v}) := (\mathbf{v}; \mathbf{x}_{-S})$, an affine map satisfying $\|A_S(\mathbf{u}) - A_S(\mathbf{v})\| = \|\mathbf{u} - \mathbf{v}\|$ (an isometry onto its image). The slice $\mathcal{C}_S = A_S^{-1}(\mathcal{C})$ is convex as the preimage of a convex set under an affine map, closed as the preimage of a closed set under a continuous map, and bounded since it embeds isometrically into the bounded set $\mathcal{C}$; hence it is convex and compact. Writing $f^S = f \circ A_S$, Lipschitz continuity is inherited via the isometry,

$$|f^S(\mathbf{u}) - f^S(\mathbf{v})| \leq L_0 \|\mathbf{u} - \mathbf{v}\|,$$

and by the chain rule $\nabla f^S(\mathbf{v}) = [\nabla f(A_S\mathbf{v})]_S$, so

$$\begin{aligned}\|\nabla f^S(\mathbf{u}) - \nabla f^S(\mathbf{v})\| &\leq \|\nabla f(A_S\mathbf{u}) - \nabla f(A_S\mathbf{v})\| \\ &\leq L_1 \|\mathbf{u} - \mathbf{v}\|.\end{aligned}$$

Moreover the coordinatewise central differences of f^S along $i \in S$ coincide with the corresponding central differences of f at $A_S(\mathbf{v})$, so the central-difference bias bound (7) and the pruning bound of Lemma 4.1 hold on $\mathcal{C}_S$ with d replaced by d_S. Lipschitz pruning therefore never discards a sub-cube of $\mathcal{C}_S$ containing a minimizer of f^S; that sub-cube's center is sampled at every stage and lies within $\delta_m \to 0$ of the minimizer, so by continuity—Proposition 4.2 applied to f^S on $\mathcal{C}_S$—the best sampled value converges to $\min_{\mathcal{C}_S} f^S$. Finally, every statement of Section 4 depends only on (L_0, L_1, μ, d), the dyadic structure, and the relevant assumptions; when Assumptions 2.1 and 2.2 hold for f^S on $\mathcal{C}_S$, Theorems B.3–B.7 instantiate verbatim with $d \to d_S$. □

Proof of Theorem B.5. Write $\Delta_m := f(\mathbf{x}_s[m]) - f(\mathbf{x}^*)$, $V_m := \|\mathbf{x}_s[m] - \mathbf{x}^*\|_2^2$, and $\tilde{\nabla} f(\mathbf{x}_s[m]) = \nabla f(\mathbf{x}_s[m]) + \mathbf{b}_m$ with finite-difference bias $\|\mathbf{b}_m\|_2 \leq \beta_m := \frac{\sqrt{d}}{2} L_1 h_m$.

Step 1 (conditional moments). Since $q_{b,m}$ is $\mathcal{F}_m$-measurable and $\mathbb{E}[\mathbf{1}(b \in S_m) \mid \mathcal{F}_m] = q_{b,m}$,

$$\begin{aligned}\mathbb{E}[\mathbf{g}_m \mid \mathcal{F}_m] &= \sum_{b=1}^{B} q_{b,m}\, q_{b,m}^{-1}\, U_b \tilde{\nabla}_{I_b} f(\mathbf{x}_s[m]) \\ &= \tilde{\nabla} f(\mathbf{x}_s[m]);\end{aligned}$$

only the marginal inclusion probabilities enter, so correlation between blocks (e.g. sampling without replacement) is immaterial. Because distinct blocks have disjoint supports, the cross terms in $\|\mathbf{g}_m\|_2^2$ vanish identically, giving

$$\begin{aligned}\mathbb{E}\big[\|\mathbf{g}_m\|_2^2 \mid \mathcal{F}_m\big] &= \sum_{b=1}^{B} \frac{\|\tilde{\nabla}_{I_b} f(\mathbf{x}_s[m])\|_2^2}{q_{b,m}} \\ &\leq \frac{\|\tilde{\nabla} f(\mathbf{x}_s[m])\|_2^2}{q_{\min}} \leq \frac{G_{\max}^2}{q_{\min}},\end{aligned}$$

using $\|\nabla f\|_2 \leq L_0$ (f is L_0-Lipschitz and differentiable by L_1-smoothness) and $h_m \leq h_1$.

Step 2 (descent recursion). Since $\mathbf{x}^* \in C'$ and $\Pi_{C'}$ is nonexpansive,

$$\begin{aligned}V_{m+1} \leq V_m &- 2\eta(m)\langle \mathbf{g}_m, \mathbf{x}_s[m] - \mathbf{x}^* \rangle \\ &+ \eta^2(m) \|\mathbf{g}_m\|_2^2.\end{aligned}$$

Taking $\mathbb{E}[\cdot \mid \mathcal{F}_m]$ and applying Step 1 (note $\mathbf{x}_s[m]$ is $\mathcal{F}_m$-measurable), then convexity of f on C', $\langle \nabla f(\mathbf{x}_s[m]), \mathbf{x}_s[m] - \mathbf{x}^* \rangle \geq \Delta_m$, and Cauchy–Schwarz, $\langle \mathbf{b}_m, \mathbf{x}_s[m] - \mathbf{x}^* \rangle \geq -\beta_m D$:

$$\begin{aligned}\mathbb{E}[V_{m+1} \mid \mathcal{F}_m] \leq V_m &- 2\eta(m)\Delta_m + 2D\,\eta(m)\beta_m \\ &+ \frac{\eta^2(m) G_{\max}^2}{q_{\min}}.\end{aligned} \tag{21}$$

Step 3 (almost-sure convergence). Under either schedule of Definition B.2, $\sum_m \eta^2(m) < \infty$ and, with $h_m = \mathcal{O}(\eta(m))$, $\sum_m \eta(m)\beta_m < \infty$. The Robbins–Siegmund almost-supermartingale theorem (Robbins & Siegmund, 1971) applied to (21) yields that V_m converges almost surely and $\sum_m \eta(m)\Delta_m < \infty$ almost surely. Since $\sum_m \eta(m) = \infty$ and $\Delta_m \geq 0$, we get $\liminf_m \Delta_m = 0$ almost surely, hence $\min_{j \leq m} f(\mathbf{x}_s[j]) \to f(\mathbf{x}^*)$ almost surely. By quadratic growth (Assumption 2.2), $\Delta_m \geq \frac{\mu}{2} V_m$; along a subsequence with $\Delta_m \to 0$ we get $V_m \to 0$, and since V_m converges almost surely its limit is zero: $\mathbf{x}_s[m] \to \mathbf{x}^*$ almost surely. (In particular $\mathbf{x}^*$ is the unique global minimizer in C': a second one would contradict the growth inequality.)

Step 4 (μ-free rate). Take total expectations in (21), sum over $m = 1, \ldots, M$, and use $\mathbb{E}V_{M+1} \geq 0$ and $V_1 \leq D^2$:

$$2 \sum_{m=1}^{M} \eta(m)\, \mathbb{E}[\Delta_m] \leq D^2 + \frac{G_{\max}^2}{q_{\min}} \sum_{m=1}^{M} \eta^2(m) + \sqrt{d}\, L_1 D \sum_{m=1}^{M} \eta(m) h_m.$$

Pathwise, $\min_{m \leq M} \Delta_m \leq \sum_m \eta(m)\Delta_m / \sum_m \eta(m)$, so the same bound holds for $\mathbb{E}[\min_{m \leq M} \Delta_m]$ after normalization. With $\eta(m) = m^{-1/2-\varepsilon}$ and $h_m = \mathcal{O}(\eta(m))$, both series in the numerator are convergent with value $\Theta(1/\varepsilon)$, while $\sum_{m \leq M} \eta(m) = \Theta(M^{1/2-\varepsilon})$; hence $\mathbb{E}[\min_{m \leq M} \Delta_m] = \mathcal{O}\big(q_{\min}^{-1} M^{-1/2+\varepsilon}\big)$ with hidden constant of order $1/(\varepsilon\, q_{\min})$.

Step 5 (μ-adapted rate). Splitting the convexity bound and inserting growth as in the proof of Theorem B.3(ii), $\langle \nabla f(\mathbf{x}_s[m]), \mathbf{x}_s[m] - \mathbf{x}^* \rangle \geq \frac{1}{2}\Delta_m + \frac{\mu}{4} V_m$, so (21) sharpens to

$$\mathbb{E}[V_{m+1} \mid \mathcal{F}_m] \leq \Big(1 - \tfrac{\mu}{2}\eta(m)\Big) V_m - \eta(m)\Delta_m + 2D\, \eta(m)\beta_m + \frac{\eta^2(m) G_{\max}^2}{q_{\min}}.$$

Write $v_m := \mathbb{E}[V_m]$ and $d_m := \mathbb{E}[\Delta_m]$, insert $\eta(m) = 4/(\mu(m+1))$ so that $1 - \frac{\mu}{2}\eta(m) = \frac{m-1}{m+1}$, and multiply through by $m(m+1)$: using $m(m+1)\eta(m) = 4m/\mu$ and $m(m+1)\eta^2(m) \leq 16/\mu^2$,

$$m(m+1) v_{m+1} \leq m(m-1) v_m - \tfrac{4m}{\mu} d_m + \tfrac{8D}{\mu} m\beta_m + \tfrac{16 G_{\max}^2}{\mu^2 q_{\min}}.$$

Summing over $m = 1, \ldots, M$ telescopes (the $m{=}1$ term vanishes) and $M(M+1)v_{M+1} \geq 0$ gives

$$\sum_{m=1}^{M} m\, d_m \leq 2D \sum_{m=1}^{M} m\beta_m + \frac{4MG^2}{\mu\, q_{\min}}.$$

Since $h_m \leq c_h/m$, $\sum_{m \leq M} m\beta_m \leq \frac{\sqrt{d}}{2} L_1 c_h M$, while $\sum_{m \leq M} m = M(M+1)/2$; hence

$$\mathbb{E}\Big[\min_{1 \leq m \leq M} \Delta_m\Big] \leq \frac{\sum_{m \leq M} m\, d_m}{\sum_{m \leq M} m} \leq \frac{2\sqrt{d}\, L_1 D\, c_h + 8G_{\max}^2/(\mu\, q_{\min})}{M+1},$$

as claimed. As in Theorem B.3, a stochastic treatment of minibatch evaluation noise is left to future work. □

Proof of Corollary 4.5. Consistency. Under the assumptions of Proposition 4.4, $\delta_n \to 0$ almost surely and Lipschitz pruning retains at every segment a cell intersecting X^*; its sampled center lies within δ_n of X^*, so by continuity the best sampled value converges to $f(\mathbf{x}^*)$ almost surely. This holds regardless of what the local stages do: the incumbent tracks the best point over both stages, and additional evaluations cannot increase it. When $X^* = \{\mathbf{x}^*\}$, convergence of the best sampled point follows by compactness and continuity as in Proposition 4.2.

Localized rate. The retained cell containing $\mathbf{x}^*$ has diameter $\delta_n \to 0$ almost surely and contains $\mathbf{x}^*$; since $\mathbf{x}^*$ is interior to $\mathcal{C}$ with $\mathbb{B}(\mathbf{x}^*, \rho) \subseteq \mathcal{C}$, almost surely there is a finite segment $n^\dagger$ such that this cell lies inside $\mathbb{B}(\mathbf{x}^*, \rho)$ for all $n \geq n^\dagger$. Any local stage run in that cell is then the iteration of Theorem B.5 with C' a convex sub-cube containing $\mathbf{x}^*$ inside the growth ball, so, conditionally on the history up to its start, conclusions (i)–(iii) of Theorem B.5 apply. □

K Additional Theoretical Discussion

This appendix collects discussion and remarks referenced from Section 4 and Section 5 that elaborate on, but are not needed to state, the paper's certified guarantees.

Theory–implementation gap (Section 4). The implemented protocol differs from the certified conceptual method in five respects: (i) a per-client transmit-gate and a client-vote accept/reject replace exact evaluation of the averaged objective; (ii) each client's uncertified local dyadic-plus-gradient search replaces smoothness-certified Lipschitz pruning: the two commit-only-if-improving fixes (Section 3) guard against regression on a single visit, but not against a locally-converged, globally-suboptimal point; (iii) the search uses finite dyadic depth and a fixed gradient-step budget rather than an asymptotic refinement schedule; (iv) permutation-based selection replaces the i.i.d. positive selection-probability floor of Theorem B.5 with sampling without replacement: within an epoch each block is drawn at most once, and only the $B \bmod B_{\mathrm{cap}}$ blocks left at the end of the permutation are skipped; and (v) activity, transmit-gate, and accept/reject messages are computed from local minibatches.

On the block partition. A sixth difference deserves its own note. Proposition 4.4 counts refinements per coordinate, so it applies although the implementation redraws the partition once per epoch (Section 3): coordinates are reshuffled within each layer and re-sliced, and an epoch skips only the $r = B \bmod B_{\mathrm{cap}}$ blocks left at the end of its permutation, so each coordinate is covered in an epoch with probability $1 - r/B > 0$, independently across epochs, which is exactly the proposition's condition. Re-randomizing is deliberate: it decorrelates which coordinates are optimized jointly, so no fixed grouping can persistently starve a direction.

On the central-difference estimator. Both the certified analysis and the implementation use the central difference $[f_k(\mathbf{x} + h\mathbf{e}_i) - f_k(\mathbf{x} - h\mathbf{e}_i)]/2h$. Under the standing L_1-Lipschitz-gradient assumption, its coordinatewise bias is at most $L_1 h/2$ (Eq. (7)), so all convergence proofs use the same error bound. For the certified partition step, the estimate is evaluated at cell centers with $h \leq \delta/(2\sqrt{d})$, which keeps both probes inside the cell. For the localized projected-refinement results, the symmetric probes are required to remain in the global feasible box $\mathcal{C}$; this is automatically possible once the retained cell lies in the interior neighborhood of an interior minimizer; before that, the local step is skipped whenever a symmetric probe is infeasible, and global consistency is unaffected, since it comes from the partition-and-refinement mechanism rather than the local stage. If bounded third derivatives are additionally available, the central-difference bias improves to $\mathcal{O}(h^2)$, but this stronger assumption is not required for any theorem in the paper.

On quantization (Section 5). Losses are nonnegative, so for loss-scale quantities the range $[0, \ln C_{\mathrm{cls}}]$, whose upper end is the cross-entropy of the uniform predictor over C_{cls} classes, covers values from random initialization onward (it is not a universal upper bound on the loss; clipping above it is innocuous in practice), giving a quantization step of $\ln(10)/256 \approx 0.009$ for 10-class problems and ≈ 0.018 for 100 classes. Transmitted points would reuse the 8-bit budget with the block's current box as the scale; as noted in Section 5, our simulations exchange values at full precision. The quantization error enters the analysis through the pruning margin $\varepsilon_{\mathrm{prune}} = \varepsilon_{\mathrm{eval}}(2 + \sqrt{d}\,\delta/(2h))$ of Lemma 4.1: because quantized losses pass through the finite-difference quotient, the error is amplified by $1/h$ per coordinate, and the margin must grow accordingly.

On Assumption 2.2. Note $\mu \leq L_1$ always; combining quadratic growth at the boundary of $\mathbb{B}(\mathbf{x}^*, \rho)$ with L_0-Lipschitzness of f likewise forces $\mu \leq 2L_0/\rho$, so μ is theoretically pinned by L_0. In practice L_0 itself may be unknown, but any μ satisfying the assumption is by construction already bounded by it, so the μ-adapted schedule needs no separate estimate of L_0.

On Theorem B.6. The constant κ^d is exponential in d and condition-number dependent; the theorem's content is that the count is *uniformly bounded over segments*, not that it is small. Under the looser

L_0-only rule of Lemma B.1 the count provably grows without bound near smooth minimizers, which is why Lemma 4.1 is required.

On Proposition 4.4. The cumulative-cost bounds of Theorem B.7 rest on the survivor-count bound of Theorem B.6, whose packing argument assumes synchronously refined cells of uniform size. Its extension to the anisotropic cells produced by blockwise refinement—in effect a near-optimality-dimension analysis under a product metric—is an open problem.

On Proposition 4.3. Proposition 4.3 alone cannot certify global optimality of f: alternating slice-global solves with frozen complements converges, in general, only to a *blockwise* optimum. Globality is restored by the jointly partitioned, Lipschitz-pruned refinement of Proposition 4.4, and, once the search has localized, by convexity: on $\mathbb{B}(\mathbf{x}^*, \rho)$, a point that is blockwise optimal for every block of a partition of the coordinates is a global minimizer of the convex restriction, since the gradient vanishes blockwise and box constraints decompose coordinatewise.

On Corollary 4.5. As with Theorem B.3, the rate clause applies to local stages run in a cell containing $\mathbf{x}^*$ (guaranteed by the every-survivor variant at a $g\kappa^d$ overhead); the consistency clause needs no such care. Two remarks delimit the result. First, the certified selector is randomized, and so is the implemented one: each epoch draws a fresh random permutation of all blocks. They differ in the sense that matters for the rate clause. Permutation sampling is without replacement, so once a block is drawn its conditional inclusion probability is zero for the rest of the epoch; it therefore does not satisfy the floor $q_{\min} > 0$ of Theorem B.5, whose rate applies to the i.i.d. selector only. What permutation sampling does provide is the coverage condition of Proposition 4.4: each block is drawn at most once per epoch, and an epoch skips only the $r = B \bmod B_{\mathrm{cap}}$ blocks left at the end of its permutation, so each coordinate is covered in an epoch with probability $1 - r/B > 0$, independently across epochs, and hence infinitely often almost surely. A deterministic top-k score rule, by contrast, has $q_{b,m} \in \{0, 1\}$ and can leave blocks untouched indefinitely. In preliminary experiments (not reported), a divergence/gradient-score-based ranking was also measurably worse than permutation sampling, because a small set of high-scoring blocks dominated selection while most of the model went untouched. Second, blockwise operation is not a query-complexity win: an iteration costs $1 + 2sd'$ rather than $2d + 1$ evaluations, but the rate carries $q_{\min}^{-1} = B/s$, so in total queries the blockwise bound is worse by a factor of order $(d/(sd'))^{1/2+\varepsilon}$; the benefit is the per-iteration cost, parallelism across blocks, and fidelity to the implemented protocol, as is standard for randomized coordinate methods (Nesterov, 2012; Richtárik & Takáč, 2014).

Accept/reject vote alignment. The combination step forms a candidate $\mathbf{x}_b^+$ from whichever clients transmitted, and the accept/reject vote decides whether $\mathbf{x}_{I_b} \leftarrow \mathbf{x}_b^+$; that vote is the only place the implemented protocol touches anything resembling the averaged objective $f = \frac{1}{K}\sum_k f_k$, since neither the local search nor the combination step ever evaluates f directly. Majority vote need not agree with the true sign of $f(\mathbf{x}_{I_b}) - f(\mathbf{x}_b^+)$ under heterogeneous data, the same failure mode known for sign-based methods with majority vote (Bernstein et al., 2018; Karimireddy et al., 2019; Chen et al., 2020). The following elementary condition delimits when the vote is provably aligned with f.

Lemma K.1 (Accept/reject alignment under bounded dissimilarity)**.** *Fix a block b with pre-update value $\mathbf{x}_{I_b}$ and combined candidate $\mathbf{x}_b^+$, let $\Gamma = f(\mathbf{x}_{I_b}) - f(\mathbf{x}_b^+)$ be the true averaged improvement, and let the client dissimilarity on their freshly sampled batches be $\zeta = \max_k \big|\, [f_k(\mathbf{x}_{I_b}) - f_k(\mathbf{x}_b^+)] - \Gamma \,\big|$. If $\zeta < |\Gamma|$, then every client's vote agrees with the sign of Γ, and hence so does the majority.*

Proof. Client k votes "improved" iff $g_k := f_k(\mathbf{x}_{I_b}) - f_k(\mathbf{x}_b^+) > 0$. Since $|g_k - \Gamma| \le \zeta < |\Gamma|$, g_k has the same sign as Γ for every k. □

When the true improvement is small relative to client dissimilarity—late in training, or under strong label skew—votes can disagree with f, and no guarantee is claimed; this is the same known tradeoff of replacing a held-out check with client consensus (Section 7).

For a probabilistic counterpart, model each client's vote as an independent, possibly noisy, measurement of the true sign of Γ.

Lemma K.2 (Majority correctness under sampled votes)**.** *Suppose the K client votes are i.i.d. Bernoulli with $\Pr(\text{client's vote matches the sign of } \Gamma) = p \ge \frac{1}{2} + \Delta$ for some $\Delta > 0$. Then the majority vote matches the sign of Γ with probability at least $1 - \exp(-2K\Delta^2)$.*

Proof. Let $X_k \in \{0,1\}$ indicate agreement with the true sign, i.i.d. Bernoulli(p), so $\mathbb{E}\sum_k X_k = pK \geq (\frac{1}{2}+\Delta)K$. The majority is wrong only if $\sum_k X_k < K/2 \leq pK - \Delta K$. Hoeffding's inequality gives $\Pr\left(\sum_k X_k - pK \leq -\Delta K\right) \leq \exp(-2\Delta^2 K)$. □

This is a simpler, two-outcome analogue of a plurality bound: accept/reject is a binary vote rather than a $2^{d'}$-way choice among children, so no union bound over competing outcomes is needed.

Neither lemma yields an infinite-horizon guarantee: a fixed per-visit error probability does not vanish over epochs. Unlike a design that permanently narrows a shared box, however, an incorrectly rejected update is not lost: ρ_0 resets fresh around $\mathbf{x}_{I_b}$ every time the block is next drawn from the next permutation, so the same (or a related) update can be proposed again on a later visit. An incorrectly *accepted* update has no such automatic correction: it persists until a later visit's local search and vote happen to reverse it, so no reversal guarantee follows in that direction either.

L Additional Benchmark Results

Section 6.1 summarizes the comparison with the zero-order baselines (Table 1, Figure 1). Table 2 gives the full statistics for all five benchmarks, and Figure 3 the trade-off curves on each of them; both also include FeedSign, discussed below. The appendix comparison also includes FeedSign (Cai et al., 2025), a federated zero-order method in which every client sends only the sign of a scalar directional derivative along a shared-seed direction, i.e., one bit per client and round in each direction. FeedSign is designed for fine-tuning pretrained models near a good optimum, whereas our benchmarks train from scratch, so this is not a one-to-one comparison: with its published step sizes FeedSign stalls or diverges from scratch, and we report it with a step size calibrated for this setting ($\eta = 0.5$, $\mu = 10^{-3}$). We include it because it marks the one-bit extreme of federated zero-order communication, and run it for as many rounds as fit in BlackBoxFL's communication budget. It remains well below BlackBoxFL on every benchmark, and on MNIST/TinyCNN and CIFAR-10/LeNet5 its accuracy first rises and then collapses to near chance, consistent with a fixed-step majority sign vote that, once its signal-to-noise ratio is too low, no longer makes net progress and can undo earlier gains. As it is not used here as intended, we report it only in this appendix.

L.1 Comparison with First-Order Methods

For reference, we also compare against three first-order communication-efficient methods: *signSGD* (Bernstein et al., 2018), *FetchSGD* (Rothchild et al., 2020), and *BiCompFL* (Egger et al., 2025). These methods compute gradients by backpropagation, so they solve an easier problem than zero-order methods and are not direct competitors; they indicate what communication and accuracy a gradient-based method achieves on the same benchmarks. BiCompFL uses the authors' original implementation and bit accounting, available for Shakespeare/NanoLM and MNIST/LeNet5. Table 3 lists the statistics and Figure 4 the trade-off curves. BlackBoxFL uses the least communication on every benchmark, including against BiCompFL (3.1 versus 14.2 Mb on Shakespeare/NanoLM and 6.29 versus 7.3 Mb on MNIST/LeNet5), and on Shakespeare/NanoLM it also attains the lowest perplexity of any method (11.3, against 11.8 for signSGD and 18.9 for BiCompFL). On the image-classification benchmarks the first-order methods generally reach higher accuracy: signSGD leads on all four, at 93–1,230× more communication.

M Models Used

M.1 Full Hyperparameter List

Unless noted otherwise, all reported runs use the default configuration referenced in Section 3: K=10 clients, block dimension d'=2, block budget B_{cap}=30, box radius ρ_0=0.5, dyadic-narrowing levels Λ=2 followed by up to 10 local gradient-descent steps ($\max_\text{grad}_\text{steps}$), gradient-descent convergence and minimum-improvement thresholds of 10^{-3} each, initial gradient step η_0=0.1 with decay c_η=0.99 and floor $\eta_{\min}$=$5{\cdot}10^{-3}$, finite-difference step h=10^{-3} (scaled by $0.1\times$ the current box width and floored at 10^{-4}), activity majority fraction ν_{act}=$\frac{1}{2}$, accept/reject majority fraction ν_{acc}=$\frac{1}{2}$, and n_q=8-bit quantized broadcasts. The activity threshold ε_{act} and

Table 2: Full comparison with the zero-order baselines: final test metric, median with 5–95% seed interval in brackets, and median cumulative communication over training ($K = 10$ clients). *Rounds* counts communication rounds (cycles for BlackBoxFL). *Fwd* is the median number of batch-level forward-pass invocations over training, in millions; batch sizes and per-method query configurations differ and are not equalized, so it is an order-of-magnitude compute indicator. No method performs backward passes. Best metric and lowest total communication per benchmark in **bold**. FeedSign is designed for fine-tuning pretrained models and is not a one-to-one comparison when training from scratch (see text).

Benchmark	Method	Rounds	Final metric	Up (Mb)	Down (Mb)	Total (Mb)	Fwd (M)
Shakespeare/NanoLM (ppl. ↓)	**BlackBoxFL (ours)**	1000	**11.3 [11.2, 11.5]**	2.76	0.37	3.13	34.2
	FedZO	1000	13.6 [13.6, 13.7]	38,441	3,844	42,285	3.0
	BAFFLE	1000	15.1 [14.9, 15.3]	48.0	3,844	3,892	1.6
	DeComFL	1000	13.1 [13.0, 13.2]	48.0	4.80	52.8	3.0
	FeedSign	272727	15.9 [15.7, 17.0]	2.73	0.27	**3.00**	5.5
MNIST/TinyCNN (acc. %)	**BlackBoxFL (ours)**	1000	**96.4 [96.0, 96.8]**	3.66	0.43	**4.09**	32.4
	FedZO	1000	90.7 [90.5, 91.2]	2,904	290	3,194	3.0
	BAFFLE	1000	92.3 [91.5, 93.4]	48.0	290	338	1.6
	DeComFL	1000	92.4 [91.5, 93.6]	48.0	4.80	52.8	3.0
	FeedSign	372074	13.8 [9.4, 40.5]	3.72	0.37	**4.09**	7.4
MNIST/LeNet5 (acc. %)	**BlackBoxFL (ours)**	1000	**95.4 [94.8, 96.2]**	5.85	0.44	6.29	32.1
	FedZO	1000	75.3 [66.1, 83.2]	14,216	1,422	15,638	3.0
	BAFFLE	1000	76.4 [66.4, 88.3]	48.0	1,422	1,470	1.6
	DeComFL	1000	75.3 [66.4, 88.0]	48.0	4.80	52.8	3.0
	FeedSign	550000	83.2 [70.9, 85.7]	5.50	0.55	**6.05**	11.0
CIFAR-10/LeNet5 (acc. %)	**BlackBoxFL (ours)**	300	**39.2 [37.9, 40.8]**	1.98	0.14	**2.11**	10.1
	FedZO	1000	29.6 [27.1, 31.3]	19,842	1,984	21,826	3.0
	BAFFLE	1000	35.3 [32.6, 37.2]	48.0	1,984	2,032	1.6
	DeComFL	1000	35.6 [32.8, 37.4]	48.0	4.80	52.8	3.0
	FeedSign	192244	10.0 [10.0, 18.2]	1.92	0.19	**2.11**	3.8
CIFAR-10/TinyCIFARWide (acc. %)	**BlackBoxFL (ours)**	3000	**53.7 [51.5, 54.1]**	20.2	1.43	**21.6**	18.9
	FedZO	3000	39.4 [38.3, 40.5]	30,530	3,053	33,583	9.1
	BAFFLE	3000	41.9 [40.7, 43.5]	144	3,053	3,197	4.7
	DeComFL	3000	43.1 [41.9, 43.9]	144	14.4	158	9.1
	FeedSign	1967900	32.0 [28.9, 36.5]	19.7	1.97	**21.6**	39.4

Table 3: Comparison with first-order communication-efficient methods, same conventions as Table 2; BlackBoxFL is repeated for reference. First-order methods additionally perform one backward pass per forward pass. †FetchSGD diverges on a subset of Shakespeare seeds (95th-percentile perplexity above 10^8); its median is shown and its curve is omitted from Figure 4. Lowest total communication per benchmark in **bold**.

Benchmark	Method	Rounds	Final metric	Up (Mb)	Down (Mb)	Total (Mb)	Fwd (M)
Shakespeare/NanoLM (ppl. ↓)	**BlackBoxFL (ours)**	1000	**11.3 [11.2, 11.5]**	2.76	0.37	**3.13**	34.2
	signSGD	1000	11.8 [11.7, 11.9]	1,201	3,844	5,045	0.07
	FetchSGD	1000	28.2†	9,600	3,844	13,444	0.07
	BiCompFL	200	18.9 [18.2, 19.7]	7.50	6.72	14.2	–
MNIST/TinyCNN (acc. %)	**BlackBoxFL (ours)**	1000	96.4 [96.0, 96.8]	3.66	0.43	**4.09**	32.4
	signSGD	1000	**98.2 [98.0, 98.3]**	90.7	290	381	0.07
	FetchSGD	1000	97.3 [96.1, 97.7]	9,600	290	9,890	0.07
MNIST/LeNet5 (acc. %)	**BlackBoxFL (ours)**	1000	95.4 [94.8, 96.2]	5.85	0.44	**6.29**	32.1
	signSGD	1000	96.8 [87.2, 97.7]	444	1,422	1,866	0.07
	FetchSGD	1000	74.2 [59.9, 97.9]	9,600	1,422	11,022	0.07
	BiCompFL	200	**99.1 [99.0, 99.2]**	3.87	3.47	7.34	–
CIFAR-10/LeNet5 (acc. %)	**BlackBoxFL (ours)**	300	39.2 [37.9, 40.8]	1.98	0.14	**2.11**	10.1
	signSGD	1000	**62.1 [60.0, 64.0]**	620	1,984	2,604	0.07
	FetchSGD	1000	41.7 [37.1, 44.1]	9,600	1,984	11,584	0.07
CIFAR-10/TinyCIFARWide (acc. %)	**BlackBoxFL (ours)**	3000	53.7 [51.5, 54.1]	20.2	1.43	**21.6**	18.9
	signSGD	3000	**75.2 [74.2, 76.9]**	954	3,053	4,007	0.2
	FetchSGD	3000	53.6 [49.0, 58.0]	28,800	3,053	31,853	0.2

transmit-gate threshold $\varepsilon_{\mathrm{tx}}$ are the only settings that differ across benchmarks: $(\varepsilon_{\mathrm{act}}, \varepsilon_{\mathrm{tx}}) = (10^{-3}, 10^{-3})$ for MNIST/TinyCNN, $(10^{-5}, 10^{-4})$ for MNIST/LeNet5, CIFAR-10/LeNet5, and CIFAR-10/TinyCIFARWide, and $(10^{-3}, 10^{-4})$ for Shakespeare/NanoLM.

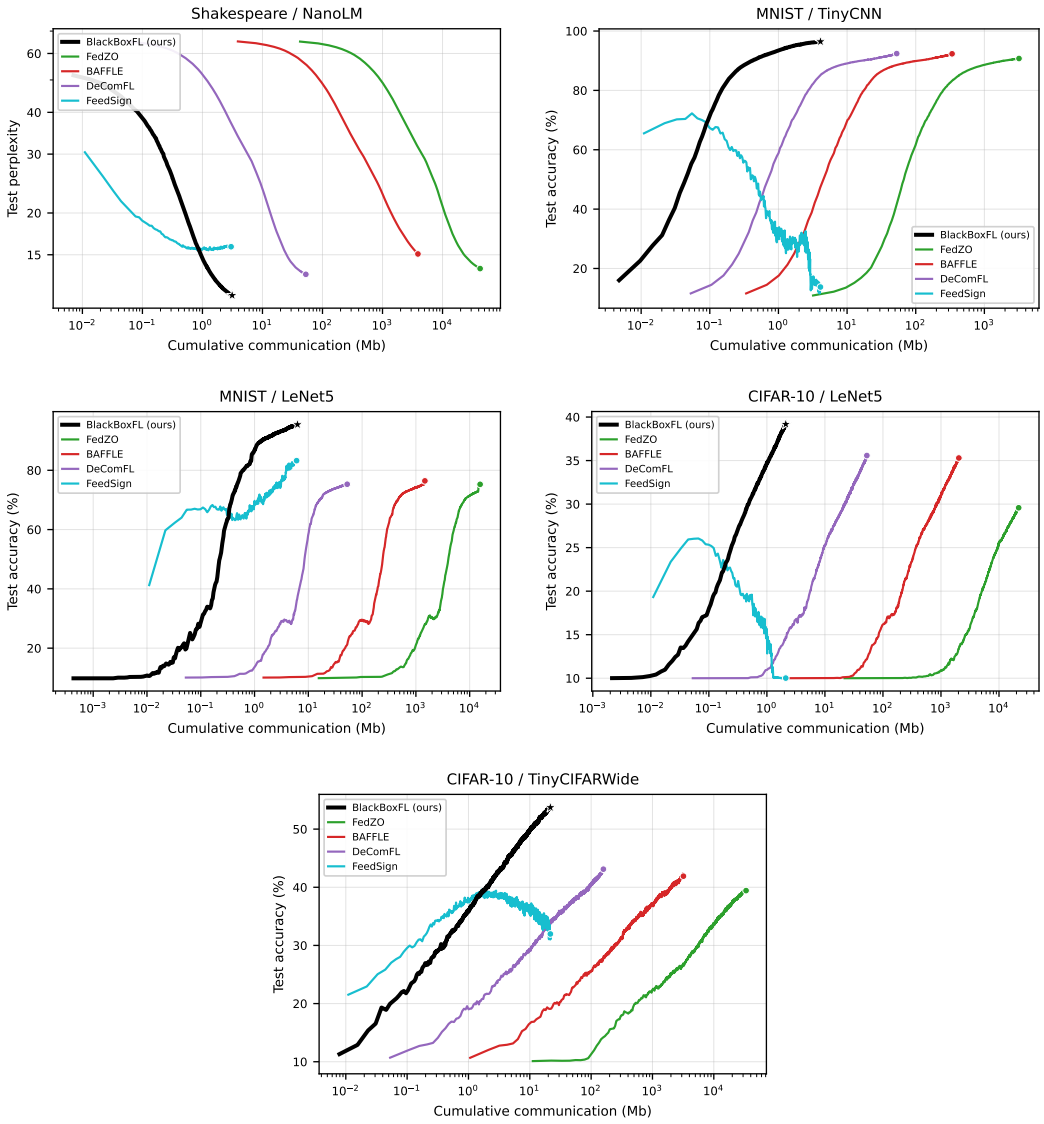


Figure 3: Metric–communication trade-off against the zero-order baselines, on all five benchmarks (median over seeds; log-scale communication, and log-scale perplexity for Shakespeare; markers show final points).

M.2 Hardware and Software

All experiments ran on a dual-GPU server with 2× NVIDIA H200 NVL (141 GB HBM3e each; NVIDIA driver 610.43.02), two 32-core AMD EPYC 9375F CPUs (128 threads), and 2 TB of RAM, under Rocky Linux 9.7. The protocol is implemented in Python 3.9 with PyTorch 2.8 (CUDA 12.8); each seed runs on a single GPU, with several seeds co-scheduled per card. Reported communication (bits) and compute (forward-pass counts) are hardware-independent: the GPU affects only wall-clock time.

M.3 MNIST

This subsection presents the architecture details of the neural network models used for MNIST digit classification. Each table shows the layer name, layer type, input dimensions, output dimensions, and number of trainable parameters. We use the ∼9k-parameter TinyCNN for the ablation studies and

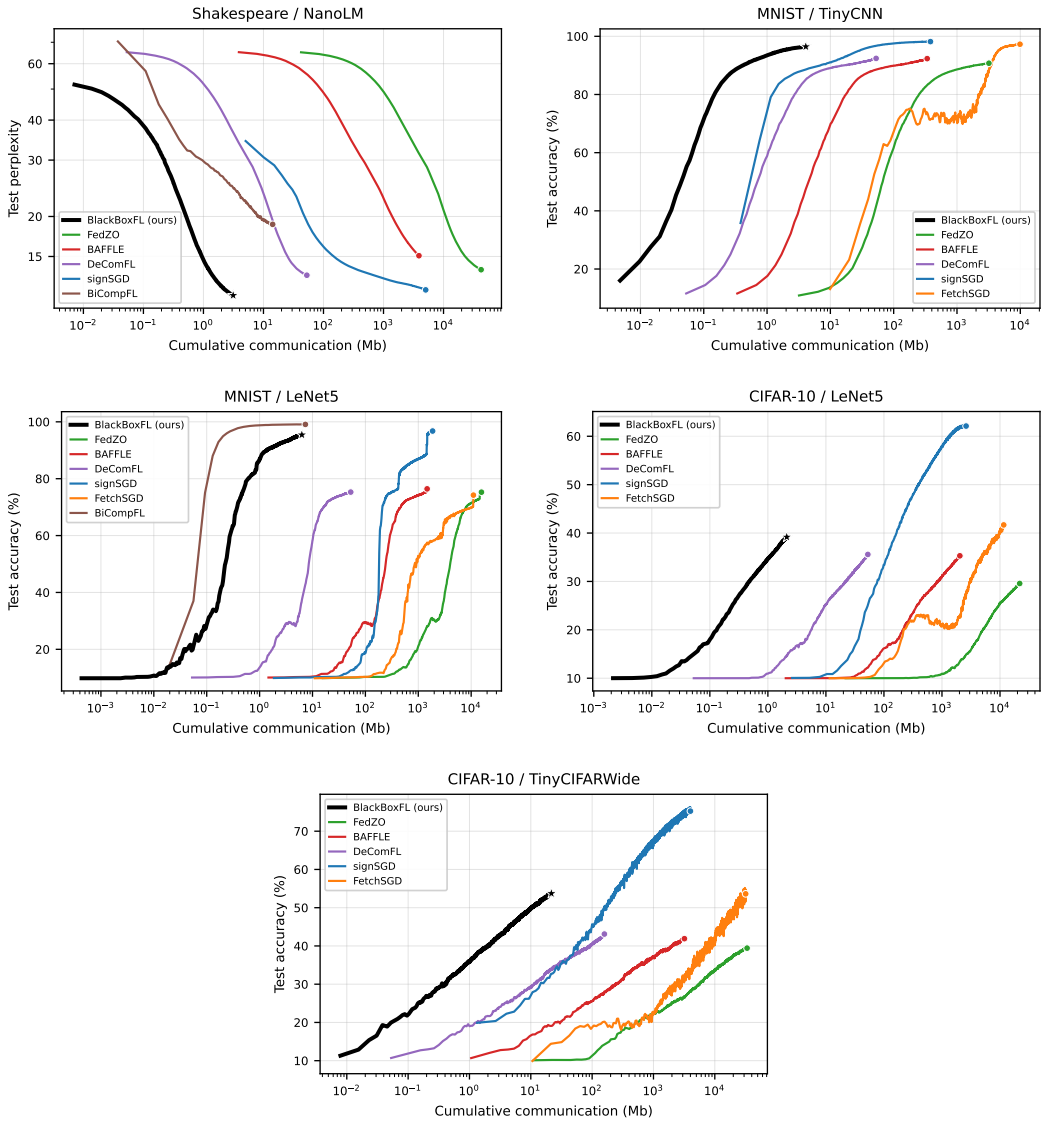


Figure 4: Metric–communication trade-off of all methods, including the first-order references (median over seeds; log-scale communication, and log-scale perplexity for Shakespeare; markers show final points).

a second MNIST head-to-head comparison, and the 44,426-parameter LeNet5 for a head-to-head comparison on MNIST.

All models expect input images of size $1 \times 28 \times 28$.

M.3.1 TinyModel — ∼10k Parameters

A minimal 2-layer CNN with roughly 9,000 parameters, see Table 4.

M.3.2 LeNet5 — ∼45k Parameters

The classic LeNet-5 architecture (Lecun et al., 1998), see Table 5.

Table 4: TinyCNN Architecture

Layer	Type	In	Out	Params
conv1	Conv2d(1→8, k=3, s=2, p=1)	1×28×28	8×14×14	72
conv2	Conv2d(8→16, k=3, s=2, p=1)	8×14×14	16×7×7	1,152
flatten	Flatten	16×7×7	784	0
fc	Linear(784→10)	784	10	7,850
			Total:	**9,074**

Table 5: LeNet5 Architecture

Layer	Type	In	Out	Params
conv1	Conv2d(1→6, k=5, s=1)	1×28×28	6×24×24	156
avgpool	AvgPool2d(k=2, s=2)	6×24×24	6×12×12	0
conv2	Conv2d(6→16, k=5, s=1)	6×12×12	16×8×8	2,416
avgpool	AvgPool2d(k=2, s=2)	16×8×8	16×4×4	0
flatten	Flatten	16×4×4	256	0
fc1	Linear(256→120)	256	120	30,840
fc2	Linear(120→84)	120	84	10,164
fc3	Linear(84→10)	84	10	850
			Total:	**44,426**

M.4 SHAKESPEARE

For the character-level language-modeling task on tiny-Shakespeare (Karpathy, 2015), we use *NanoLM*, a small decoder-only Transformer: 2 pre-LayerNorm blocks, 2 attention heads, model width $d_{\text{model}} = 64$, feed-forward width $d_{\text{ff}} = 256$, GELU activations, a 256-token causal context window, and a 65-character vocabulary, with the output projection weight-tied to the token embedding; see Table 6.

M.5 CIFAR-10

The CIFAR-10 head-to-head comparison uses a LeNet5 adapted to a $3\times32\times32$ RGB input, with $62{,}006$ trainable parameters, see Table 7. It is BatchNorm-free by design: because BlackBoxFL is purely zero-order (forward-pass loss evaluations only—never a train-mode forward or a backward pass), BatchNorm running statistics would never update, so a plain convolutional backbone is used for an apples-to-apples comparison with the MNIST/LeNet5 setup. All CIFAR-10 methods share this model and return raw logits.

The second CIFAR-10 comparison uses TinyCIFARWide, a wider GroupNorm-and-global-average-pooling CNN with $31{,}802$ trainable parameters, see Table 8. GroupNorm, unlike BatchNorm, keeps no running statistics, so it behaves identically under the forward-only evaluation of zero-order methods. All methods share this model and are run for $3{,}000$ rounds.

N ABLATION STUDIES

In this section we study how various hyperparameters affect our algorithm. Unless noted otherwise, ablations are conducted on the TinyCNN model with i.i.d. MNIST data (the data-heterogeneity study

Table 6: NanoLM Architecture

Layer	Type	Params
tok_emb	Embedding($65 \to 64$), tied with head	4,160
pos_emb	Embedding($256 \to 64$)	16,384
blocks $\times 2$	Causal self-attn (2 heads) + GELU MLP ($64 \to 256 \to 64$), pre-LN	99,456
ln_f	LayerNorm(64)	128
head	Linear($64 \to 65$), tied with tok_emb	0
	Total (unique):	**120,128**

Table 7: LeNet5-CIFAR Architecture (CIFAR-10, $3\times32\times32$ input)

Layer	Type	In	Out	Params
conv1	Conv2d(3→6, k=5, s=1)	$3\times32\times32$	$6\times28\times28$	456
avgpool	AvgPool2d(k=2, s=2)	$6\times28\times28$	$6\times14\times14$	0
conv2	Conv2d(6→16, k=5, s=1)	$6\times14\times14$	$16\times10\times10$	2,416
avgpool	AvgPool2d(k=2, s=2)	$16\times10\times10$	$16\times5\times5$	0
flatten	Flatten	$16\times5\times5$	400	0
fc1	Linear(400→120)	400	120	48,120
fc2	Linear(120→84)	120	84	10,164
fc3	Linear(84→10)	84	10	850
			Total:	**62,006**

uses Dirichlet partitions), a 300-cycle budget, and 10 seeds per configuration; plots show the median test accuracy over completed seeds as a function of cumulative communication.

N.1 Number of Clients

We sweep $K \in \{2, 5, 10, 20, 50\}$ and report test accuracy as a function of cumulative communication in Figure 5. Consistent with Equation (13), the per-cycle communication cost scales with K: a larger client count reaches a given accuracy at proportionally higher total communication, so fewer clients are more communication-efficient in this full-participation, deterministic-evaluation setup—additional clients contribute vote redundancy rather than gradient-noise reduction—while the final accuracy is comparable across client counts (all within 91–95%). Measured *per cycle*, however, a larger client count converges *faster* and to a slightly higher final accuracy (Figure 6): $K{=}50$ and $K{=}20$ reach 95.0% within 300 cycles, while $K{=}5$ reaches 93.4% and $K{=}2$ lags at 91.0%. A larger client count is therefore not merely redundant: it buys faster convergence, traded against its proportionally higher communication.

N.2 Data Heterogeneity (Non-IID)

We study robustness to non-IID client data using a Dirichlet(α) label-skew partition, where a smaller α produces more heterogeneous clients. Figure 7 reports test accuracy as a function of cumulative communication for $\alpha \in \{0.3, 1.0\}$ against the IID setting ($\alpha \to \infty$). BlackBoxFL degrades gracefully: final accuracy falls from 94.2% (IID) to 92.4% at $\alpha = 1.0$ and 86.3% at $\alpha = 0.3$.

N.3 Data Heterogeneity on CIFAR-10

We repeat the data-heterogeneity study on CIFAR-10 (LeNet5, Dirichlet(α) label skew, $\alpha \in \{0.3, 1.0\}$ against the IID split) to test whether the graceful degradation seen on MNIST transfers to a harder task (Figure 8). It transfers only partially: final accuracy falls from 38.0% (IID)

Table 8: TinyCIFARWide Architecture (CIFAR-10, 3×32×32 input)

Layer	Type	In	Out	Params
conv1	Conv2d(3→20, k=3, p=1) + GroupNorm(4) + ReLU	3×32×32	20×32×32	600
avgpool	AvgPool2d(k=2, s=2)	20×32×32	20×16×16	0
conv2	Conv2d(20→40, k=3, p=1) + GroupNorm(4) + ReLU	20×16×16	40×16×16	7,320
avgpool	AvgPool2d(k=2, s=2)	40×16×16	40×8×8	0
conv3	Conv2d(40→64, k=3, p=1) + GroupNorm(8) + ReLU	40×8×8	64×8×8	23,232
gap	AdaptiveAvgPool2d(1) + Flatten	64×8×8	64	0
fc	Linear(64→10)	64	10	650
			Total:	**31,802**

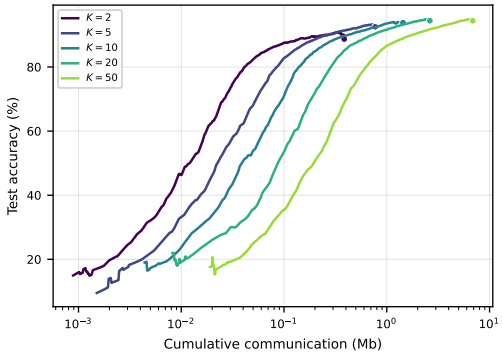


Figure 5: Number-of-clients ablation: test accuracy as a function of cumulative communication for $K \in \{2, 5, 10, 20, 50\}$ (TinyCNN, block size $d' = 2$).

to 31.7% at $\alpha = 1.0$ and 24.2% at $\alpha = 0.3$—relative drops of 17% and 36%, versus 2% and 8% on MNIST—so heterogeneity costs substantially more on the harder task, consistent with the accept/reject vote-alignment failure mode of Lemma K.1: as client dissimilarity ζ grows relative to the true improvement Γ, the majority vote stops tracking the averaged objective.

N.4 Communication Budget (B_{cap})

The number of blocks refined per cycle, B_{cap}, is the primary communication–accuracy knob: a larger B_{cap} spends more bits per cycle (Equation (13)) but converges in fewer rounds. Figure 9 traces this trade-off for $B_{\text{cap}} \in \{5, 15, 30, 60, 100\}$ on TinyCNN. All budgets lie on essentially the same accuracy-per-bit frontier; B_{cap} controls how far along that frontier training travels within a fixed number of cycles, from 86.6% at $B_{\text{cap}} = 5$ to 96.4% at $B_{\text{cap}} = 100$, with diminishing returns beyond $B_{\text{cap}} = 30$.

N.5 Block Size (d')

The block size d' sets the dimensionality of each local search and therefore the per-block report cost: a transmitting client sends $n_q(d' + 1)$ bits, and its local search costs $2d'$ forward passes per gradient estimate (Section 3), while a smaller d' yields more, smaller blocks ($B \approx d/d'$) for a fixed model dimension d. We sweep $d' \in \{1, 2, 3, 4, 5\}$ on TinyCNN, holding $B_{\text{cap}} = 30$, $\Lambda = 2$, and $\varepsilon_{\text{act}} = 10^{-3}$ fixed, and report test accuracy as a function of cumulative communication in Figure 10. As with B_{cap}, all block sizes lie on essentially the same accuracy-per-bit curve: a larger d' spends more bits per cycle and so travels further along it within the fixed budget, from 91.3% at $d' = 1$ (0.6 Mb) to 96.1% at $d' = 5$ (4.1 Mb).

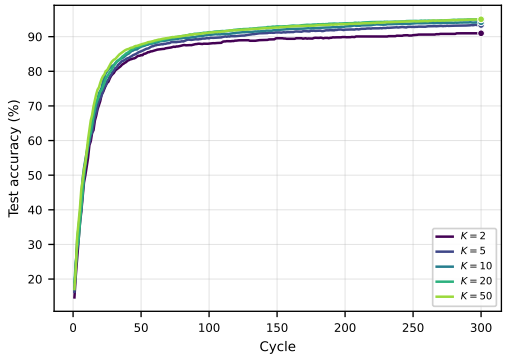


Figure 6: Convergence rate vs. number of clients: test accuracy as a function of *cycle* for $K \in \{2, 5, 10, 20, 50\}$ (TinyCNN). Complements Figure 5 (vs. communication): more clients converge faster per round and reach slightly higher accuracy, so the client count trades communication for convergence speed.

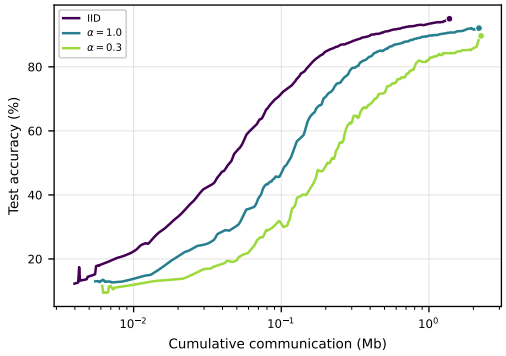


Figure 7: Data-heterogeneity ablation: test accuracy as a function of cumulative communication for BlackBoxFL under Dirichlet(α) non-IID label skew on TinyCNN, versus the IID split. Smaller α is more heterogeneous.

N.6 Activity Threshold

The activity pre-screen skips a block when its local zero-order gradient norm falls below ε_{act}, trading communication against the risk of stalling. Its effect is *model-dependent*: on the compact TinyCNN, accuracy is essentially flat (93.9–94.6%) for $\varepsilon_{\text{act}} \leq 5{\cdot}10^{-3}$. At the default 10^{-3} screening barely changes communication (1.33 Mb, the same as without screening, $\varepsilon_{\text{act}} = 0$); at $5{\cdot}10^{-3}$ it cuts it by about a fifth (1.04 Mb), and at 10^{-2} it halves it (0.63 Mb) but final accuracy drops to 92.5%, whereas on larger models the threshold must be lowered to avoid screening out *every* block and stalling training. Figure 11 reports test accuracy as a function of cumulative communication for a range of ε_{act}.

N.7 Component Study

We remove one mechanism of the protocol at a time: the transmit-gate (every client always reports), the accept/reject vote (every combined candidate is committed), and the activity screen (every selected block is searched); we also replace the two-phase local search by dyadic narrowing alone ($\Lambda = 4$ levels, no gradient steps). Figure 12 reports the results on TinyCNN. The transmit-gate is the main source of communication savings: without it, total communication nearly doubles (2.34 versus 1.29 Mb) for a similar final accuracy (94.9% versus 94.1%). The other variants change neither accuracy nor communication appreciably on this compact model (94.3–94.8%, 1.27–1.39 Mb).

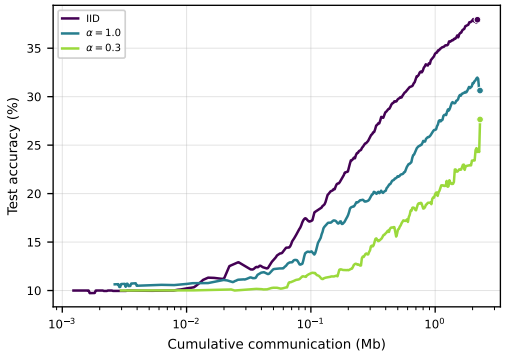


Figure 8: Data-heterogeneity ablation on CIFAR-10 (LeNet5): test accuracy as a function of cumulative communication under Dirichlet(α) non-IID label skew, versus the IID split.

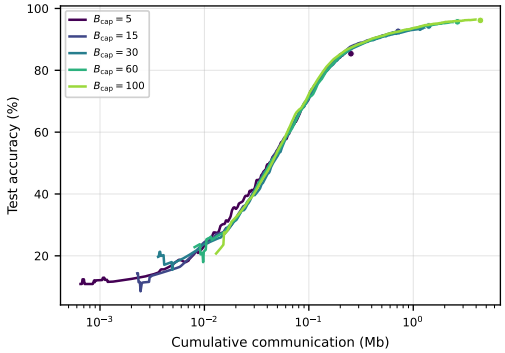


Figure 9: Communication-budget ablation: test accuracy as a function of cumulative communication for different numbers of refined blocks B_{cap} per cycle (TinyCNN). Larger B_{cap} advances faster along the communication axis.

N.8 Update Dynamics

Figure 13 reports, for the MNIST/LeNet5 configuration at the default settings (median over 100 seeded runs), how the 30 selected blocks are dispatched each cycle. Averaged over runs and training, 2.2 blocks per cycle are dropped at the activity pre-screen and a further 1.4 receive no client report at all (every client declined at the transmit-gate, so the block is a no-op and no vote is held). The remaining $\approx$26.4 blocks reach an accept/reject vote, which commits 17.1 and rolls back 9.3 of them per cycle, a 65% accept rate. The transmit-gate is the busier filter at the client level: 27% of all client–block pairs decline to report, which is where most of the per-cycle payload saving in Section 5 comes from. About a third of voted blocks are thus rejected after their reports were already paid for.

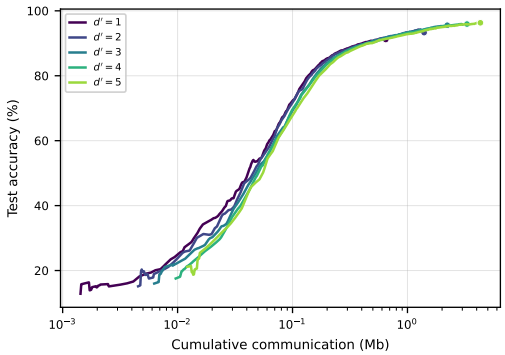


Figure 10: Block-size ablation: test accuracy as a function of cumulative communication for $d' \in \{1, 2, 3, 4, 5\}$ (TinyCNN, $B_{\text{cap}} = 30$). Smaller d' sends fewer uplink bits per report but requires more rounds to cover d dimensions; larger d' makes each block's local search higher-dimensional.

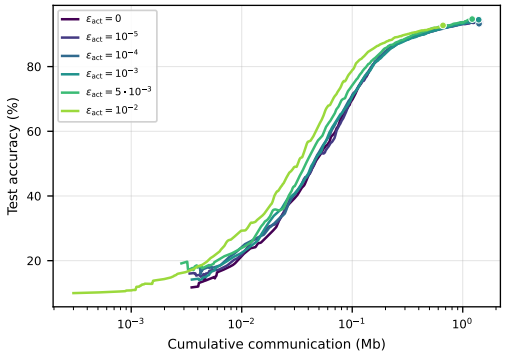


Figure 11: Activity-threshold ablation on TinyCNN: test accuracy as a function of cumulative communication for a range of pre-screen thresholds ε_{act}; the curves nearly coincide up to $\varepsilon_{\text{act}} = 5{\cdot}10^{-3}$; at 10^{-2} training stops earlier on the communication axis at lower accuracy. The threshold is a model-dependent knob (see text).

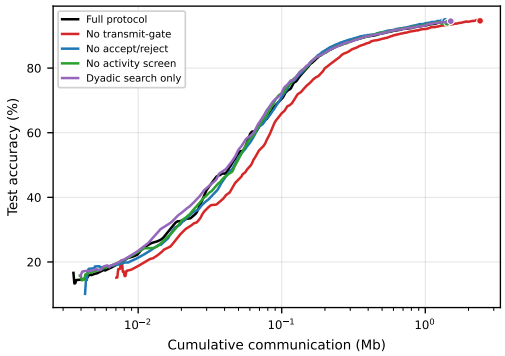


Figure 12: Component study on TinyCNN: test accuracy as a function of cumulative communication for the full protocol and for variants with the transmit-gate, the accept/reject vote, or the activity screen removed, or with dyadic narrowing as the only local search.

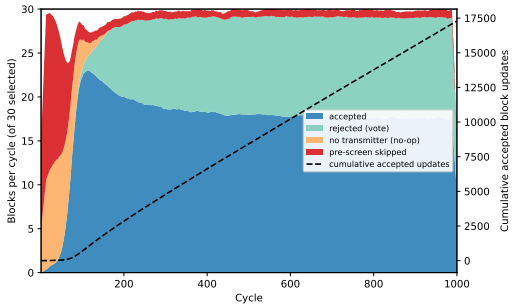


Figure 13: Update dynamics on MNIST/LeNet5 (median over 100 runs): per-cycle dispatch of the 30 selected blocks (accepted, rejected at the vote, no-transmitter no-ops, pre-screen-skipped; left axis, smoothed) and the cumulative count of accepted block updates (right axis).