UNDERSTANDING TOP-k SPARSIFICATION IN DISTRIBUTED DEEP LEARNING WITH LAPLACEK

Anonymous authors
Paper under double-blind review

ABSTRACT

Top-k sparsification reduces the communication cost of distributed stochastic gradient descent, but existing convergence analyses often characterize its error using the looser Random-k contraction. We analyze error-compensated gradients and show that their coordinate magnitudes exhibit a Laplace-like structure, which leads to a tighter characterization of the error introduced by Top-k selection. Based on this observation, we introduce LaplaceK, a GPU-oriented selector that uses a Laplace-tail prior to localize a small candidate pool, performs exact Top-k selection within that pool, and falls back to global selection when the candidate pool is invalid. This design preserves the exact selected coordinates and transmitted values while reducing selection overhead. Experiments across six non-Transformer image-model configurations show that LaplaceK maintains convergence and final accuracy comparable to magnitude-based sparsification while reducing wall-clock training time. Across the reported experiments, LaplaceK achieves up to $1.70\times$ speedup over standard TopK-SGD.

1 INTRODUCTION

Distributed stochastic gradient descent (SGD) reduces training time by distributing computation across many workers, but synchronization can become the main bottleneck as models and clusters scale Dean et al. (2012); Goyal et al. (2017); Jia et al. (2018); You et al. (2020). In data-parallel training, workers must exchange increasingly large updates, which can limit scalability when fast accelerators are connected by limited networks Strom (2015). Gradient compression reduces this overhead through quantization, which lowers the bitwidth of transmitted values, and sparsification, which transmits only selected coordinates Seide et al. (2014); Alistarh et al. (2017); Wen et al. (2017); Aji & Heafield (2017); Wangni et al. (2018). Quantization introduces approximation noise, whereas sparsification may bias updates when important coordinates are omitted. Error accumulation and error feedback mitigate these effects by retaining discarded information across iterations Seide et al. (2014); Karimireddy et al. (2019). Error-compensated Top-k sparsification is particularly attractive because it communicates the largest-magnitude coordinates while storing omitted updates in a local residual Lin et al. (2018). These properties motivate a coupled statistical and systems question: can the nonuniform structure of training gradients be exploited to reduce selection cost without changing the exact Top-k operator?

To investigate this question, one must consider both the optimization behavior and the computational cost of Top-k. The standard analysis treats Top-k through the same generic contraction used for Random-k Stich et al. (2018); Alistarh et al. (2018); Karimireddy et al. (2019). Although this abstraction provides a useful distribution-free convergence guarantee, it does not capture how gradient magnitudes affect the quality of a selected update. It therefore cannot explain why TopK-SGD and Random-k stochastic gradient descent (RandomK-SGD) may behave differently at the same sparsity level, or how error feedback interacts with nonuniform gradient energy. Related work studies gradient sparsification, biased compressors, and acceleration under more specific models Wangni et al. (2018); Beznosikov et al. (2023); Li et al. (2020), while sketching provides an alternative for reducing communication Ivkin et al. (2019). At the systems level, exact Top-k selection requires global ordering or repeated selection over a high-dimensional gradient Shanbhag et al. (2018); Fang et al. (2019). Approximate methods therefore use sampled thresholds, trimmed searches, or related candidate-reduction strategies Lin et al. (2018); Fang et al. (2019). The resulting objective is to exploit gradient structure to reduce selection overhead while preserving exact Top-k semantics.

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.

We meet this objective by identifying a recurring Laplace-like structure in error-compensated gradients and using it to reduce the computational cost of graphics processing unit (GPU)-based Top-k selection. We then develop the corresponding theoretical analysis and evaluate the proposed method across different models, sparsity levels, and worker configurations. Our contributions are summarized as follows:

1. We identify a Laplace-like gradient distribution across different model architectures and tasks, and propose LaplaceK, a GPU-oriented selector that constructs a bounded candidate pool, performs exact Top-k selection within the pool, and uses global fallback to preserve exact Top-k semantics.
2. We establish contraction bounds for Top-k, derive a sharper bound under an ordered-tail condition, and prove the convergence of LaplaceK under standard stochastic optimization assumptions. We also characterize Top-k error under a Laplace model and compare it with Random-k.
3. We evaluate LaplaceK on different model configurations. The results show that LaplaceK preserves optimization quality while improving training efficiency, with additional evidence from gradient-distribution diagnostics, ablations, and multi-worker experiments.

2 PRELIMINARIES

Let P be the number of workers, $\boldsymbol{x}_t \in \mathbb{R}^d$ the model parameters, and $\boldsymbol{g}_t^p$ the stochastic gradient computed by worker p at iteration t. Distributed synchronous stochastic gradient descent (SGD) applies

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_t \frac{1}{P} \sum_{p=1}^{P} \boldsymbol{g}_t^p, \tag{1}$$

where η_t is the learning rate. A k-sparse compressor Comp_k reduces communication by retaining at most k coordinates. With error feedback Stich et al. (2018); Karimireddy et al. (2019), each worker forms

$$\boldsymbol{u}_t^p = \boldsymbol{g}_t^p + \boldsymbol{\epsilon}_t^p, \qquad \boldsymbol{\epsilon}_0^p = \mathbf{0}, \tag{2}$$

updates the model using

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_t \frac{1}{P} \sum_{p=1}^{P} \mathrm{Comp}_k(\boldsymbol{u}_t^p), \tag{3}$$

and stores the omitted coordinates:

$$\boldsymbol{\epsilon}_{t+1}^p = \boldsymbol{u}_t^p - \mathrm{Comp}_k(\boldsymbol{u}_t^p). \tag{4}$$

For $\boldsymbol{x} \in \mathbb{R}^d$, $\mathrm{Top}_k(\boldsymbol{x})$ retains the k largest coordinates in magnitude, whereas $\mathrm{Rand}_k(\boldsymbol{x})$ samples k coordinates uniformly. Top-k minimizes the discarded energy among all k-coordinate selections:

$$\|\boldsymbol{x} - \mathrm{Top}_k(\boldsymbol{x})\|^2 \le \|\boldsymbol{x} - \mathrm{Rand}_k(\boldsymbol{x})\|^2. \tag{5}$$

For Random-k,

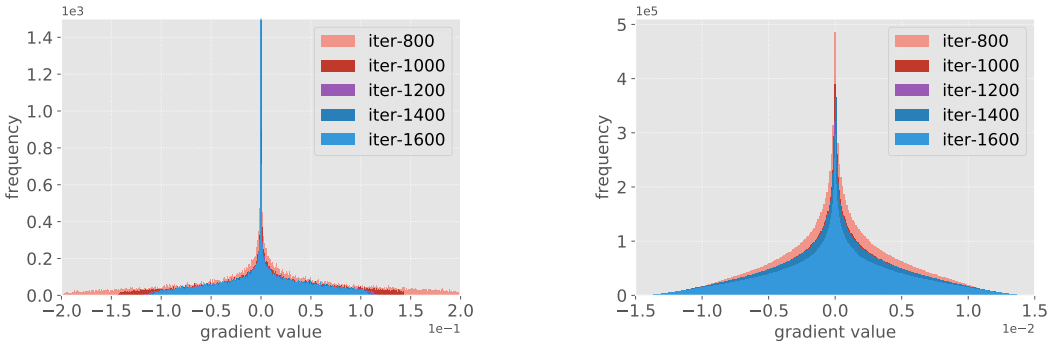
$$\mathbb{E}_R\left[\|\boldsymbol{x} - \mathrm{Rand}_k(\boldsymbol{x})\|^2\right] = \left(1 - \frac{k}{d}\right) \|\boldsymbol{x}\|^2. \tag{6}$$

Consequently, both compressors satisfy the generic contraction

$$\mathbb{E}_C\left[\|\boldsymbol{x} - \mathrm{Comp}_k(\boldsymbol{x})\|^2\right] \le \left(1 - \frac{k}{d}\right) \|\boldsymbol{x}\|^2. \tag{7}$$

This common bound provides a distribution-free convergence benchmark, while the additional energy retained by magnitude-based selection is characterized by the distributional analysis.

Figure 1 shows averaged error-compensated gradient histograms for LeNet-5 and the Visual Geometry Group network (VGG-16) under Top-k stochastic gradient descent (TopK-SGD). Both models exhibit a sharp central peak near zero together with substantial, nearly symmetric two-sided tails, suggesting a Laplace-like structure. Additional model configurations and gradient-distribution observations are



(a) LeNet-5 (b) Visual Geometry Group network (VGG-16)

Figure 1: Averaged coordinate histograms of the error-compensated gradient under Top-k stochastic gradient descent (TopK-SGD) for LeNet-5 and the Visual Geometry Group network (VGG-16). The sharp central peak and substantial symmetric tails motivate the Laplace-like prior used for threshold localization. The overlays correspond to sampled training iterations.

reported in Appendix C.2. This structure motivates using the Laplace prior to localize the Top-k threshold. Exact Top-k selection still requires global operations on the full gradient Shanbhag et al. (2018); Fang et al. (2019), whereas sampled and trimmed threshold searches can reduce this cost Lin et al. (2018); Fang et al. (2019). LaplaceK exploits this structure to localize the tail efficiently, perform exact selection within a bounded candidate pool, and retain a global fallback that preserves exact Top-k semantics.

3 LaplaceK: Laplace-Prior Candidate Selection for Top-K

Guided by the observations in Figure 1, LaplaceK separates statistical localization from exact selection. A Laplace-tail prior constructs a small candidate pool, exact Top-K is applied only within that pool, and a global fallback preserves correctness across search outcomes. The LaplaceK preserves the sparsity level and selected values across prior configurations through its exact selection and fallback behavior, while the prior guides the search cost.

Let $g_t \in \mathbb{R}^N$ denote the gradient at iteration t and let $e_t \in \mathbb{R}^N$ denote the residual accumulated from previous iterations. Before sparsification, LaplaceK forms

$$x_t = g_t + e_t. \tag{8}$$

Given density $d \in (0, 1]$, the target number of transmitted coordinates is

$$K = \max\left(1, \lfloor dN \rfloor\right). \tag{9}$$

LaplaceK selects an index set I_t with $|I_t| = K$ and transmits $(x_{t,i})_{i \in I_t}$ together with the corresponding indices. The residual is then updated by

$$e_{t+1,i} = \begin{cases} 0, & i \in I_t, \\ x_{t,i}, & i \notin I_t. \end{cases} \tag{10}$$

This error-feedback rule accumulates omitted entries for reconsideration in subsequent iterations.

LaplaceK computes the Laplace scale of x_t using one Compute Unified Device Architecture (CUDA) reduction. For a zero-centered Laplace prior, the maximum-likelihood scale parameter is the mean absolute coordinate value,

$$b_t = \frac{1}{N} \sum_{i=1}^{N} |x_{t,i}|. \tag{11}$$

The method uses the zero-centered Laplace tail as a computational prior for the magnitude distribution. For the target tail probability

$$\alpha = \frac{K}{N}, \tag{12}$$

Algorithm 1 LaplaceK with error feedback

Require: Gradient tensor g with at least one element, tensor identifier `name`, density $\rho \in [0, 1]$, pool ratio $R \geq 1$, integer threshold budget $T \geq 0$, residual map `residual`
Ensure: Selected global indices I and signed values V

if `residual[name]` does not exist or has a changed shape **then**
  `residual[name]` $\leftarrow \text{zeroslike}(g)$
end if
$x \leftarrow \text{flatten}(g) + \text{flatten}(\texttt{residual[name]})$
$N \leftarrow \text{numel}(x)$, $K \leftarrow \min(\max(\lfloor N\rho \rfloor, 1), N)$
$\alpha \leftarrow K/N$ {Retained fraction; Equation (12)}
$b \leftarrow N^{-1} \sum_i |x_i|$ {Laplace scale; Equation (11)}
`valid_pool` $\leftarrow$ `false`
if $b < 10^{-8}$ **then**
  $I \leftarrow \text{ExactTopK}(|x|, K)$
else
  $z \leftarrow -\log(\max(\alpha, 10^{-12}))$ {Laplace tail; Equation (13)}
  `valid_pool` $\leftarrow$ `false`
  for $j = 1$ to $\min(T, 100)$ **do**
    $\tau \leftarrow \max(bz, 10^{-10})$ {Threshold; Equation (13)}
    $C \leftarrow \{i : |x_i| \geq \tau\}$, $c \leftarrow |C|$ {Candidate set; Equation (14)}
    if $K \leq c \leq \min(N, RK)$ **then**
      `valid_pool` $\leftarrow$ `true`; **break**
    else if $c < K$ **then**
      $z \leftarrow 0.5z$ {Lower threshold; Equation (15)}
    else
      $z \leftarrow 0.8(c/K)z$ {Raise threshold; Equation (15)}
    end if
    $z \leftarrow \max(z, 10^{-5})$
  end for
  if `valid_pool` **then**
    $C \leftarrow \{i : |x_i| \geq \tau\}$
    if $|C| \geq K$ **then**
      $J \leftarrow \text{ExactTopK}(|x[C]|, K)$
      $I \leftarrow C[J]$ {Exact Top-K within the pool}
    else
      `valid_pool` $\leftarrow$ `false`
    end if
  end if
  if `valid_pool` $=$ `false` **then**
    $I \leftarrow \text{ExactTopK}(|x|, K)$ {Global fallback}
  end if
end if
$V \leftarrow x[I]$
$r \leftarrow \text{flatten}(x)$
Set $r[I] \leftarrow 0$ {Store omitted coordinates; Equation (10)}
UPDATERESIDUAL: `residual[name]` $\leftarrow \text{reshape}(r, \text{shape}(g))$
return (I, V)

the initial standardized threshold is set to

$$z_0 = -\log\big(\max(\alpha, 10^{-12})\big), \qquad T_0 = \max(b_t z_0, 10^{-10}). \tag{13}$$

Under the Laplace prior, the model predicts that K coordinates satisfy $|x_{t,i}| \geq T_0$, because $\mathbb{P}(|X| \geq b_t z_0) = e^{-z_0}$. This threshold produces a small candidate pool for the subsequent exact operation.

LaplaceK introduces a candidate-pool ratio $R \geq 1$, where the desired pool size lies between K and $\min(N, RK)$. The default implementation uses $R = 5$. At each threshold-search iteration, the CUDA kernel scans the vector and counts

$$C_t = |\{i : |x_{t,i}| \geq \tau_t\}| . \tag{14}$$

The threshold is adjusted according to the following multiplicative rule:

$$z_{t+1} = \begin{cases} 0.5 z_t, & C_t < K, \\ 0.8\, z_t (C_t/K), & C_t > \min(N, RK), \\ z_t, & K \leq C_t \leq \min(N, RK). \end{cases} \tag{15}$$

At iteration t, LaplaceK recomputes the physical threshold as $\tau_t = \max(b_t z_t, 10^{-10})$, while enforcing $z_t \geq 10^{-5}$ after each update. Let $C_t = \big|\{i : |x_{t,i}| \geq \tau_t\}\big|$. If $C_t < K$, the threshold is reduced; if $C_t > \min(N, RK)$, it is increased. The search stops when $K \leq C_t \leq \min(N, RK)$ or when the iteration budget is exhausted. For a valid threshold, the CUDA implementation uses warp ballots and atomic allocation to compact the qualifying indices, applies exact Top-K selection to the candidate values, maps the selected positions back to the original vector, and preserves their signs. Algorithm 1 summarizes this procedure; gray rows denote the Laplace-prior threshold, candidate construction, and reduced-pool selection steps. Let $\mathrm{ExactTopK}(v, K)$ return the indices of the K largest entries of v under a fixed tie-breaking rule. If $P_\tau = \{i : |x_i| \geq \tau\}$, $|P_\tau| \geq K$, then all coordinates outside P_τ have magnitude below τ, whereas the K-th largest magnitude is at least τ. Exact Top-K selection within P_τ therefore matches global selection. For an invalid pool or a near-zero scale, LaplaceK falls back to global exact Top-K; hence, under exact arithmetic, $x \mapsto \mathrm{Top}_K(x)$ for every input. The Laplace prior changes only the selection path and cost: after a successful search, final selection uses at most $\min(N, RK)$ candidates, with the benefit determined by density, R, GPU architecture, and fallback frequency.

4 Theoretical Analysis

We first establish the unconditional finite-dimensional contraction of exact Top-k, which applies to LaplaceK because both its candidate-pool and fallback paths are exact. We then derive a sharper ordered-tail bound and analyze the large-dimensional error ratio under an independent and identically distributed (i.i.d.) Laplace model, progressing from generic Top-k guarantees to convergence and distribution-dependent analysis. Define the retained-coordinate fraction

$$\alpha = \frac{k}{d} \in (0, 1). \tag{16}$$

This ratio will be used to express both the compression error and the convergence coefficients.

For a nonzero vector $\boldsymbol{u}$, let $\pi_{(1)} \geq \cdots \geq \pi_{(d)} \geq 0$ be the coordinates of $|\boldsymbol{u}|/\|\boldsymbol{u}\|_\infty$ in decreasing order. Then

$$\frac{\|\boldsymbol{u} - \mathrm{Top}_k(\boldsymbol{u})\|^2}{\|\boldsymbol{u}\|^2} = \frac{\sum_{i=k+1}^{d} \pi_{(i)}^2}{\sum_{i=1}^{d} \pi_{(i)}^2}. \tag{17}$$

This identity expresses the discarded-energy ratio through the ordered coordinate magnitudes and provides the basis for the contraction bounds below.

Theorem 4.1. *For every $\boldsymbol{u} \in \mathbb{R}^d$ and integer $1 \leq k \leq d$,*

$$\|\boldsymbol{u} - \mathrm{Top}_k(\boldsymbol{u})\|^2 \leq \left(1 - \frac{k}{d}\right) \|\boldsymbol{u}\|^2 = (1 - \delta_{\mathrm{topk}}) \|\boldsymbol{u}\|^2, \qquad \delta_{\mathrm{topk}} = \frac{k}{d}. \tag{18}$$

For $k = d$, the left-hand side is zero and the same statement holds with $\delta_{\mathrm{topk}} = 1$. Proof in Appendix B.1.

This theorem provides the distribution-free contraction baseline for exact Top-k. Since LaplaceK returns exact Top-k along both its candidate-pool and fallback paths, the same guarantee applies to LaplaceK for every input.

Theorem 4.2. *For a nonzero vector $\boldsymbol{u}$, suppose its ordered squared coordinates satisfy the explicit ordered-tail condition*

$$\sum_{i=k+1}^{d} \pi_{(i)}^2 \le (1-\alpha)^2 \sum_{i=1}^{d} \pi_{(i)}^2, \qquad \alpha = \frac{k}{d}. \tag{19}$$

Then

$$\|\boldsymbol{u} - \mathrm{Top}_k(\boldsymbol{u})\|^2 \le (1-\alpha)^2 \|\boldsymbol{u}\|^2 = (1-\delta_{\mathrm{quad}})\|\boldsymbol{u}\|^2, \qquad \delta_{\mathrm{quad}} = \alpha(2-\alpha). \tag{20}$$

Proof in Appendix B.3.

The ordered-tail condition captures additional concentration in the largest coordinates and yields a sharper contraction coefficient than the generic bound. This coefficient will be used below to derive a corresponding optimization guarantee. To connect these compressor bounds to stochastic optimization, we use the smooth nonconvex setting of Karimireddy et al. (2019). The following assumption isolates the smoothness and stochastic-gradient conditions required by the error-feedback analysis.

Assumption 4.3. *The objective $f : \mathbb{R}^d \to \mathbb{R}$ is L-smooth and bounded below by $f^\star := \inf_x f(x) > -\infty$. The stochastic gradient is unbiased, and its conditional second moment is bounded: for some $G > 0$,*

$$\mathbb{E}[\boldsymbol{g}_t \mid \boldsymbol{x}_t] = \nabla f(\boldsymbol{x}_t), \qquad \mathbb{E}[\|\boldsymbol{g}_t\|^2 \mid \boldsymbol{x}_t] \le G^2. \tag{21}$$

These conditions provide the standard optimization framework for analyzing error-feedback updates. In particular, they allow the compressor coefficients established above to be inserted directly into the convergence analysis.

Theorem 4.4 (Convergence of LaplaceK)**.** *Suppose Assumption 4.3 holds and Algorithm 1 is run with exact comparisons and a fixed tie-breaking rule. Then the algorithm implements the exact Top-k compressor, and with $\eta_t = 1/\sqrt{T+1}$ the error-feedback guarantee gives*

$$\min_{t\in[T]} \mathbb{E}\big[\|\nabla f(\boldsymbol{x}_t)\|^2\big] \le \frac{4(f(\boldsymbol{x}_0) - f^\star) + LG^2}{2\sqrt{T+1}} + \frac{4L^2G^2(1-\delta_{\mathrm{topk}})}{\delta_{\mathrm{topk}}^2(T+1)}. \tag{22}$$

Proof in Appendix B.2.

The theorem transfers the standard error-feedback convergence guarantee to LaplaceK through its exact selection and global fallback mechanism. We next replace the generic Top-k coefficient with the sharper ordered-tail coefficient to quantify the benefit of additional coordinate concentration.

Theorem 4.5. *Suppose Assumption 4.3 holds and every nonzero compressed input satisfies the ordered-tail condition in Theorem 4.2; a zero input trivially satisfies the contraction inequality. With $\eta_t = 1/\sqrt{T+1}$, the error-feedback guarantee of Karimireddy et al. (2019) gives*

$$\min_{t\in[T]} \mathbb{E}\big[\|\nabla f(\boldsymbol{x}_t)\|^2\big] \le \frac{4(f(\boldsymbol{x}_0) - f^\star) + LG^2}{2\sqrt{T+1}} + \frac{4L^2G^2(1-\delta_{\mathrm{quad}})}{\delta_{\mathrm{quad}}^2(T+1)}. \tag{23}$$

Proof in Appendix B.4.

This result shows how the ordered-tail structure improves the compression-dependent term in the convergence bound. We now turn to the Laplace model to quantify the threshold behavior and the energy retained by the largest coordinates.

Assumption 4.6. *For the vector under consideration, $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_d \overset{\text{i.i.d.}}{\sim} \mathrm{Laplace}(0, b)$, where $b > 0$.*

This assumption provides an explicit model for the nearly symmetric two-sided gradient tails observed empirically. Under this model, we first derive the threshold associated with a target retained fraction.

Lemma 4.7. *For*

$$z_\alpha = -\log(\alpha), \qquad \tau_\alpha = b z_\alpha, \tag{24}$$

we have $\mathbb{P}(|\boldsymbol{u}_i| \ge \tau_\alpha) = \alpha$. Proof in Appendix B.5.

This lemma shows that τ_α selects a fraction α of the coordinates under the Laplace model. It therefore provides the probabilistic foundation for LaplaceK's initial threshold.

Lemma 4.8. *Under Assumption 4.6, the empirical scale used by LaplaceK,*

$$\widehat{b}_d = \frac{1}{d}\sum_{i=1}^{d} |\boldsymbol{u}_i|, \tag{25}$$

satisfies $\widehat{b}_d \xrightarrow{\text{a.s.}} b$. *If* $k_d/d \to \alpha \in (0,1)$, *then the unclipped empirical threshold* $\widehat{b}_d[-\log(k_d/d)]$ *converges almost surely to* $\tau_\alpha = b[-\log(\alpha)]$. *Proof in Appendix B.6.*

The lemma justifies the scale computation implemented by LaplaceK: as the dimension grows, the single-reduction value converges to the population scale and yields an asymptotically calibrated threshold. We next use this threshold to compute the energy contained below the selected tail.

Lemma 4.9. *Under Assumption 4.6,*

$$\mathbb{E}\left[\boldsymbol{u}_i^2 \mathbf{1}\{|\boldsymbol{u}_i| \le \tau_\alpha\}\right] = 2b^2\left[1 - \alpha\left(1 + z_\alpha + \frac{z_\alpha^2}{2}\right)\right]. \tag{26}$$

Proof in Appendix B.7.

This closed-form expression gives the model-implied squared energy discarded below the Laplace threshold. Combined with the total second moment, it leads directly to the asymptotic Top-k error ratio.

Theorem 4.10. *Let* $k = k_d$ *satisfy* $k_d/d \to \alpha \in (0,1)$. *Under Assumption 4.6,*

$$\frac{\|\boldsymbol{u} - \mathrm{Top}_{k_d}(\boldsymbol{u})\|^2}{\|\boldsymbol{u}\|^2} \xrightarrow[d\to\infty]{\text{a.s.}} \gamma_{\text{laplace}} := 1 - \alpha\left(1 + z_\alpha + \frac{z_\alpha^2}{2}\right). \tag{27}$$

Proof in Appendix B.8.

This theorem converts the Laplace coordinate model into a quantitative prediction for the normalized Top-k error in high dimensions. We now compare this prediction with the corresponding error ratio of Random-k. For Random-k, the discarded-energy ratio is $\gamma_{\text{rand}} = 1 - \alpha$. The Laplace model predicts a strictly smaller ratio.

Lemma 4.11. *For every* $\alpha \in (0,1)$, $\gamma_{\text{laplace}} < \gamma_{\text{rand}}$. *Proof in Appendix B.9.*

This lemma formally establishes that the Laplace-like tail structure yields more retained squared energy than uniform Random-k selection at the same retained fraction. The next theorem places the Laplace prediction relative to the sharper ordered-tail benchmark.

Theorem 4.12. *For every* $\alpha \in (0,1)$,

$$\gamma_{\text{laplace}} < (1-\alpha)^2 = \gamma_{\text{quad}}. \tag{28}$$

Proof in Appendix B.10.

This comparison shows that the asymptotic Laplace error ratio is also below the quadratic ordered-tail benchmark. The resulting coefficients summarize the three compression regimes used throughout the analysis:

$$\delta_{\text{rand}} = \alpha, \qquad \delta_{\text{quad}} = \alpha(2-\alpha), \qquad \delta_{\text{laplace}} = \alpha\left(1 + z_\alpha + \frac{z_\alpha^2}{2}\right). \tag{29}$$

The first two are finite-dimensional contraction coefficients under their respective conditions, whereas the third is the asymptotic Laplace coefficient that motivates LaplaceK's threshold initialization.

5 Experiments

We evaluate three questions: whether LaplaceK preserves optimization quality under aggressive sparsification, how its candidate-selection hyperparameters affect accuracy and cost, and whether the reduced selection workload improves scaling across multiple graphics processing units (GPUs). Unless stated otherwise, each plotted training curve and summary measurement reports the mean over repeated runs. We compare dense stochastic gradient descent (Dense-SGD), Top-k stochastic

gradient descent (TopK-SGD), Random-k stochastic gradient descent (RandomK-SGD), RadiK-SGD, AirTopK-SGD, Dragonn-SGD, and LaplaceK-SGD. The magnitude-based methods use error feedback, whereas RandomK-SGD samples the sparsity pattern uniformly. LaplaceK changes only the graphics processing unit (GPU) selection path: it preserves the requested cardinality through exact candidate selection or a global fallback. The evaluation therefore considers both convergence per epoch and time to accuracy.

5.1 End-to-end comparison

Figure 2 reports validation accuracy against epochs and wall-clock time for six non-Transformer model configurations. LaplaceK-SGD remains competitive with the magnitude-based sparse baselines in final accuracy, while RandomK-SGD converges more slowly in the reported averages. The wall-clock curves show that LaplaceK can reach the high-accuracy regime much earlier without changing the communicated cardinality. LaplaceK delivers competitive optimization quality together with consistently favorable training time across the reported panels. Table 1 summarizes the measured training time. In these averaged results, LaplaceK is the fastest method for all six models. Relative to the fastest non-LaplaceK baseline for each model, it reduces wall-clock time by 3.5%–41.0%, with the largest reduction on VGG-16. These averaged measurements support a systems-level time-to-accuracy benefit, with competitive final accuracy across the reported configurations.

5.2 Ablation Study

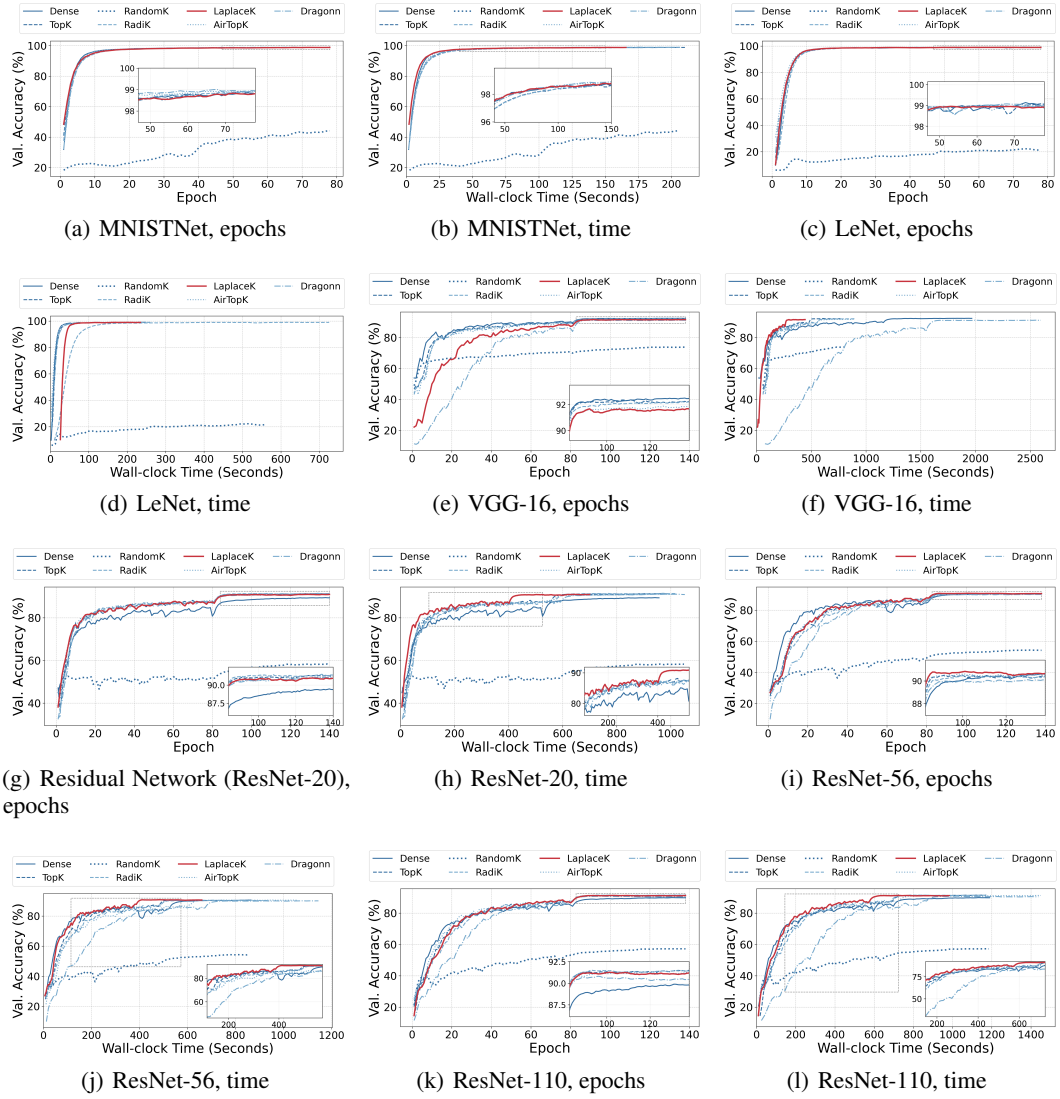
We ablate the retained density and candidate-pool ratio, the two parameters that most directly control LaplaceK's compression and selection cost. Figure 3(a) varies the retained density from 0.1 to 0.001, corresponding to nominal value compression from $10\times$ to $1000\times$. All settings reach a similar final-accuracy range in the averaged results, including density 0.001. This result is consistent with error feedback, which accumulates omitted coordinates for later iterations. The ablation characterizes mean behavior across densities and reports the corresponding accuracy profiles. Figure 3(b) varies the candidate-pool ratio and reports mean per-step compute time. The mean cost is non-monotonic over the tested range, showing that the ratio controls the systems operating point independently of the accuracy parameter. The lowest mean step time occurs at $10K$ in this sweep, and the tested ratios have similar costs.

Table 1: Mean wall-clock training time in seconds across repeated runs. Here SGD denotes stochastic gradient descent. The best mean time in each row is shown in bold.

Model	Dense SGD	TopK SGD	RandomK SGD	RadiK SGD	LaplaceK SGD	AirTopK SGD	Dragonn SGD
LeNet	248.36	251.63	559.75	725.49	**239.59**	257.69	259.70
MNISTNet	204.12	210.34	206.06	206.11	**170.09**	204.72	206.54
ResNet-20	956.43	993.95	1048.67	1017.18	**704.99**	1002.19	1051.86
ResNet-56	864.51	818.89	848.71	938.88	**674.21**	1055.23	1145.42
ResNet-110	1189.09	1172.05	1185.68	1167.01	**1002.95**	1450.30	1438.03
VGG-16	1965.75	768.06	807.76	905.82	**452.93**	890.19	2595.98

6 Conclusion

We studied Top-k sparsification from optimization and systems perspectives. Our analysis shows that Laplace-like gradient structure yields tighter error characterizations than Random-k. LaplaceK exploits this structure to construct a bounded candidate pool, perform exact Top-k selection, and use global fallback when necessary. It therefore preserves the exact Top-k indices and values while reducing selection cost. Experiments across six non-Transformer models show that LaplaceK maintains competitive optimization quality and reduces wall-clock time by 3.5%–41.0% relative to the fastest non-LaplaceK baseline, with up to $1.70\times$ speedup.



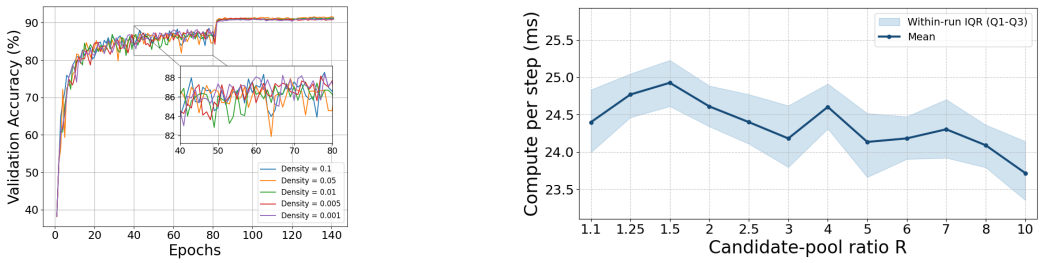
(a) MNISTNet, epochs (b) MNISTNet, time (c) LeNet, epochs

(d) LeNet, time (e) VGG-16, epochs (f) VGG-16, time

(g) Residual Network (ResNet-20), epochs (h) ResNet-20, time (i) ResNet-56, epochs

(j) ResNet-56, time (k) ResNet-110, epochs (l) ResNet-110, time

Figure 2: Mean accuracy versus epochs and wall-clock time across six model configurations, averaged over repeated runs. Here VGG-16 denotes the Visual Geometry Group network, and ResNet denotes a residual network. LaplaceK-SGD matches the optimization quality of magnitude-based sparsifiers while improving time to accuracy.



(a) Retained density (b) Candidate-pool ratio

Figure 3: Ablations of LaplaceK's retained density and candidate-pool ratio. Panel (a) reports validation accuracy across densities; panel (b) reports mean per-step compute time across pool ratios. Each setting reports the mean over repeated runs.

ARTIFICIAL INTELLIGENCE (AI) USE STATEMENT

We used generative artificial intelligence (AI) tools solely to assist with proofreading and polishing the language of this manuscript, including improving grammar, clarity, readability, and overall presentation. The tools were not used to generate research ideas, design experiments, generate synthetic datasets, formulate or prove mathematical claims, or produce substantive scientific content. All AI-assisted text was carefully reviewed, revised, and verified by the authors. The authors take full responsibility for the content and claims of this paper.

REFERENCES

Alejandro Acero. *Acoustical and environmental robustness for automatic speech recognition*. PhD thesis, Carnegie Mellon University, 1990.

Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 440–445, 2017.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pp. 1709–1720, 2017.

Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems*, pp. 5973–5983, 2018.

Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations. In *Advances in Neural Information Processing Systems*, pp. 14668–14679, 2019.

Aleksandr Beznosikov, Samuel Horváth, Peter Richtárik, and Mher Safaryan. On biased compression for distributed learning. *Journal of Machine Learning Research*, 24(276):1–50, 2023. URL https://jmlr.org/papers/v24/21-1548.html.

Chia-Yu Chen, Jungwook Choi, Daniel Brand, Ankur Agrawal, Wei Zhang, and Kailash Gopalakrishnan. AdaComp: Adaptive residual gradient compression for data-parallel distributed training. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc V. Le, and Andrew Y. Ng. Large scale distributed deep networks. In *Advances in neural information processing systems*, pp. 1223–1231, 2012.

Jiarui Fang, Haohuan Fu, Guangwen Yang, and Cho-Jui Hsieh. RedSync: Reducing synchronization bandwidth for distributed deep learning training system. *Journal of Parallel and Distributed Computing*, 133:30–39, 2019.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Farzin Haddadpour, Mohammad Mahdi Kamani, Mehrdad Mahdavi, and Viveck Cadambe. Trading redundancy for communication: Speeding up distributed SGD for non-convex optimization. In *International Conference on Machine Learning*, pp. 2545–2554, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Vladimir Braverman, Ion Stoica, and Raman Arora. Communication-efficient distributed SGD with sketching. In *Advances in Neural Information Processing Systems*, pp. 13144–13154, 2019.

Xianyan Jia, Shutao Song, Wei He, Yangzihao Wang, Haidong Rong, Feihu Zhou, Liqiang Xie, Zhenyu Guo, Yuanzhou Yang, Liwei Yu, et al. Highly scalable deep learning training system with mixed-precision: Training ImageNet in four minutes. *arXiv preprint arXiv:1807.11205*, 2018.

Peng Jiang and Gagan Agrawal. A linear speedup analysis of distributed deep learning with sparse and quantized communication. In *Advances in Neural Information Processing Systems*, pp. 2530–2541, 2018.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning*, pp. 3252–3261, 2019.

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

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Zhize Li, Dmitry Kovalev, Xun Qian, and Peter Richtárik. Acceleration for compressed gradient descent in distributed and federated optimization. In *International Conference on Machine Learning*, 2020.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*, 2018. URL https://arxiv.org/abs/1712.01887.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330, 1993.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *International Conference on Learning Representations*, 2018.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *Fifteenth Annual Conference of the International Speech Communication Association*, pp. 1058–1062, 2014. doi: 10.21437/Interspeech.2014-274.

Anil Shanbhag, Holger Pirk, and Samuel Madden. Efficient top-k query processing on massively parallel hardware. In *Proceedings of the 2018 International Conference on Management of Data*, pp. 1557–1570. ACM, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems*, pp. 4452–4463, 2018.

Nikko Strom. Scalable distributed DNN training using commodity GPU cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

Hanlin Tang, Chen Yu, Xiangru Lian, 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, 2019.

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. PowerSGD: Practical low-rank gradient compression for distributed optimization. In *Advances in Neural Information Processing Systems*, pp. 14236–14245, 2019.

Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems*, pp. 1306–1316, 2018.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems*, pp. 1509–1519, 2017.

Jiaxiang Wu, Weidong Huang, Junzhou Huang, and Tong Zhang. Error compensated quantized SGD and its applications to large-scale distributed optimization. *International Conference on Machine Learning*, 2018.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1904.00962.

APPENDIX CONTENTS

A More Related Work **15**

A.1 Quantized communication and error feedback 15

A.2 Gradient sparsification and convergence 15

A.3 Top-k selection and gradient distributions 15

B Complete Proofs **15**

B.1 Proof of Theorem 4.1 15

B.2 Proof of Theorem 4.4 16

B.3 Proof of Theorem 4.2 16

B.4 Proof of Theorem 4.5 16

B.5 Proof of Lemma 4.7 16

B.6 Proof of Lemma 4.8 16

B.7 Proof of Lemma 4.9 16

B.8 Proof of Theorem 4.10 17

B.9 Proof of Lemma 4.11 17

B.10 Proof of Theorem 4.12 17

C More Experiments **17**

C.1 Top-k versus Random-k 17

C.2 Gradient-coordinate distributions 18

C.3 Laplace versus Gaussian prior 20

C.4 Implementation Diagnostics 20

A More Related Work

A.1 Quantized communication and error feedback

Gradient quantization reduces communication by representing each coordinate with fewer bits. Early systems used 1-bit updates Seide et al. (2014), while TernGrad and Quantized Stochastic Gradient Descent (QSGD) developed ternary and stochastic low-bit compressors with optimization guarantees Wen et al. (2017); Alistarh et al. (2017). Mixed-precision training further demonstrated that reduced numerical precision can preserve model quality when scaling and accumulation are handled carefully Micikevicius et al. (2018). Error-feedback methods retain the local compression residual and reintroduce it at later iterations, correcting the persistent bias of aggressive compressors Wu et al. (2018); Karimireddy et al. (2019); Tang et al. (2019). Related analyses combine quantization, sparsification, and local computation to expose communication and convergence trade-offs Jiang & Agrawal (2018); Haddadpour et al. (2019); Basu et al. (2019).

A.2 Gradient sparsification and convergence

Sparsification removes coordinates rather than reducing their precision. Threshold and magnitude-based schemes communicate only large updates Strom (2015); Aji & Heafield (2017); Chen et al. (2018), while Deep Gradient Compression combines momentum correction, local clipping, and residual accumulation to support very high sparsity Lin et al. (2018). General convergence analyses model these methods through contractive or biased compressors Wangni et al. (2018); Stich et al. (2018); Alistarh et al. (2018); Beznosikov et al. (2023); Li et al. (2020). These frameworks establish broad guarantees, while distribution-dependent analysis further characterizes the energy advantage of Top-k over Random-k. Sketching and low-rank communication provide complementary structured alternatives Ivkin et al. (2019); Vogels et al. (2019); they reduce communication without explicitly selecting the largest coordinates.

A.3 Top-k selection and gradient distributions

The systems cost of sparsification follows both the selector and the communicated cardinality. Exact Top-k carries substantial graphics processing unit (GPU) computation because it requires global ordering or repeated selection over a high-dimensional vector Shanbhag et al. (2018). Deep Gradient Compression (DGC) obtains a threshold from a sample Lin et al. (2018), whereas RedSync uses a trimmed search to reduce selection overhead Fang et al. (2019). These approaches motivate approximate or candidate-based selectors whose efficiency follows the quality of threshold localization. Empirical studies of neural-network activations and gradients often report coordinate distributions concentrated near zero Glorot & Bengio (2010); Micikevicius et al. (2018). Our analysis uses this observation at two levels. The theoretical model quantifies the energy retained in the tails under an explicit Laplace assumption, while LaplaceK uses a Laplace tail only as a computational prior. Exact selection inside the candidate pool and a global fallback separate statistical threshold localization from the correctness of the final Top-k result.

B Complete Proofs

B.1 Proof of Theorem 4.1

Proof. Let $a_{(1)} \geq \cdots \geq a_{(d)} \geq 0$ be the sorted magnitudes of $\boldsymbol{u}$. For $k < d$, every discarded magnitude is at most $a_{(k)}$, and $ka_{(k)}^2 \leq \sum_{i=1}^{k} a_{(i)}^2$. Therefore

$$\sum_{i=k+1}^{d} a_{(i)}^2 \leq (d-k)a_{(k)}^2 \leq \frac{d-k}{k} \sum_{i=1}^{k} a_{(i)}^2. \tag{30}$$

Writing $S_{\text{keep}} = \sum_{i=1}^{k} a_{(i)}^2$ and $S_{\text{drop}} = \sum_{i=k+1}^{d} a_{(i)}^2$, the preceding inequality gives $kS_{\text{drop}} \leq (d-k)S_{\text{keep}}$. Hence

$$\frac{S_{\text{drop}}}{S_{\text{keep}} + S_{\text{drop}}} \leq \frac{d-k}{d} = 1 - \frac{k}{d}, \tag{31}$$

which is the claimed bound. If $k = d$, no coordinate is discarded and both sides of the error inequality are zero. □

B.2 Proof of Theorem 4.4

Proof. The candidate-pool lemma and the global fallback imply that the mathematical output of Algorithm 1 is $\mathrm{Top}_k(\boldsymbol{u}_t + \boldsymbol{e}_t)$ at every update. Theorem 4.1 therefore supplies the contractive-compressor condition with $\delta = \delta_{\mathrm{topk}}$. Assumption 4.3 supplies the smoothness, unbiasedness, lower-bound, and bounded-second-moment conditions required by Remark 4 of Karimireddy et al. (2019). Substituting $\eta_t = 1/\sqrt{T+1}$ and $\delta = \delta_{\mathrm{topk}}$ into that result yields Eq. 22. □

B.3 Proof of Theorem 4.2

Proof. By Eq. 17 and the ordered-tail condition in Eq. 19,

$$\frac{\|\boldsymbol{u} - \mathrm{Top}_k(\boldsymbol{u})\|^2}{\|\boldsymbol{u}\|^2} = \frac{\sum_{i=k+1}^{d} \pi_{(i)}^2}{\sum_{i=1}^{d} \pi_{(i)}^2} \le (1-\alpha)^2. \tag{32}$$

Multiplying by the positive denominator $\|\boldsymbol{u}\|^2$ gives the claimed bound. Finally, $1 - (1-\alpha)^2 = \alpha(2-\alpha) = \delta_{\mathrm{quad}}$, which gives the contraction form in Eq. 20. □

B.4 Proof of Theorem 4.5

Proof. Theorem 4.2 gives, for every nonzero compressed input,

$$\|\boldsymbol{u} - \mathrm{Top}_k(\boldsymbol{u})\|^2 \le (1 - \delta_{\mathrm{quad}})\|\boldsymbol{u}\|^2. \tag{33}$$

For a zero input, the same inequality holds with both sides equal to zero. Thus Top-k satisfies the contractive-compressor condition used in Remark 4 of Karimireddy et al. (2019), with coefficient $\delta = \delta_{\mathrm{quad}}$. Assumption 4.3 supplies the smoothness, unbiasedness, and bounded-second-moment conditions required by that result. Substituting $\eta_t = 1/\sqrt{T+1}$ and $\delta = \delta_{\mathrm{quad}}$ into the cited bound yields Eq. 23. □

B.5 Proof of Lemma 4.7

Proof. Under Assumption 4.6, $|\boldsymbol{u}_i|$ is exponential with scale b. Therefore,

$$\mathbb{P}(|\boldsymbol{u}_i| \ge \tau_\alpha) = \exp(-\tau_\alpha/b) = \exp(-z_\alpha) = \alpha, \tag{34}$$

where the last equality follows from Eq. 24. □

B.6 Proof of Lemma 4.8

Proof. For $\boldsymbol{u}_i \sim \mathrm{Laplace}(0, b)$, the variable $|\boldsymbol{u}_i|$ has finite mean and $\mathbb{E}|\boldsymbol{u}_i| = b$. The strong law of large numbers applied to the i.i.d. sequence $|\boldsymbol{u}_i|$ therefore gives $\widehat{b}_d \to b$ almost surely. If $k_d/d \to \alpha \in (0,1)$, continuity of $-\log(\cdot)$ on $(0,1)$ gives $-\log(k_d/d) \to -\log(\alpha)$. The product convergence theorem then yields $\widehat{b}_d[-\log(k_d/d)] \to b[-\log(\alpha)] = \tau_\alpha$ almost surely. □

B.7 Proof of Lemma 4.9

Proof. Using symmetry and the change of variables $t = x/b$,

$$\mathbb{E}\big[\boldsymbol{u}_i^2 \mathbf{1}\{|\boldsymbol{u}_i| \le \tau_\alpha\}\big] = b^2 \int_0^{z_\alpha} t^2 e^{-t}\, dt. \tag{35}$$

Two integrations by parts yield

$$\int_0^{z_\alpha} t^2 e^{-t}\, dt = 2 - e^{-z_\alpha}(z_\alpha^2 + 2z_\alpha + 2). \tag{36}$$

Using $e^{-z_\alpha} = \alpha$ proves Eq. 26. □

B.8 Proof of Theorem 4.10

Proof. Let $Y_1, \dots, Y_d$ denote the absolute coordinates and write their order statistics as $Y_{(1)} \le \cdots \le Y_{(d)}$. For all sufficiently large d, $d - k_d \ge 1$; define $q_d = Y_{(d-k_d)}$. Since Y_i has the continuous, strictly increasing distribution function $F(y) = 1 - \exp(-y/b)$ on $[0, \infty)$ and $(d - k_d)/d \to 1 - \alpha$, the strong law applied at any two points on either side of $\tau_\alpha = F^{-1}(1 - \alpha)$ gives $q_d \to \tau_\alpha$ almost surely. Because the distribution is continuous, ties occur with probability zero. We now justify the corresponding convergence of the trimmed second moments. For any $\varepsilon > 0$, almost surely and for all sufficiently large d, $|q_d - \tau_\alpha| < \varepsilon$, and hence

$$\begin{aligned} &\left| \frac{1}{d} \sum_{i=1}^{d} Y_i^2 \mathbf{1}\{Y_i \le q_d\} - \frac{1}{d} \sum_{i=1}^{d} Y_i^2 \mathbf{1}\{Y_i \le \tau_\alpha\} \right| \\ &\qquad \le \frac{1}{d} \sum_{i=1}^{d} Y_i^2 \mathbf{1}\{\tau_\alpha - \varepsilon < Y_i \le \tau_\alpha + \varepsilon\}. \end{aligned} \tag{37}$$

The strong law applied to the right-hand side, followed by $\varepsilon \downarrow 0$, makes the right-hand side converge to zero because $\mathbb{P}(Y_i = \tau_\alpha) = 0$ and $\mathbb{E}[Y_i^2] < \infty$. Since there are no ties, the leftmost sum is exactly the energy discarded by Top-k_d. Thus the normalized discarded energy satisfies

$$\frac{1}{d} \|\boldsymbol{u} - \mathrm{Top}_{k_d}(\boldsymbol{u})\|^2 \xrightarrow{\text{a.s.}} \mathbb{E}\left[\boldsymbol{u}_i^2 \mathbf{1}\{|\boldsymbol{u}_i| \le \tau_\alpha\}\right]. \tag{38}$$

The strong law of large numbers also gives

$$\frac{1}{d} \|\boldsymbol{u}\|^2 \xrightarrow{\text{a.s.}} \mathbb{E}[\boldsymbol{u}_i^2] = 2b^2, \tag{39}$$

The denominator limit is positive. Dividing the two limits and applying Lemma 4.9 yields

$$\frac{\|\boldsymbol{u} - \mathrm{Top}_{k_d}(\boldsymbol{u})\|^2}{\|\boldsymbol{u}\|^2} \xrightarrow{\text{a.s.}} 1 - \alpha \left(1 + z_\alpha + \frac{z_\alpha^2}{2}\right),$$

as required. □

B.9 Proof of Lemma 4.11

Proof. For $\alpha \in (0, 1)$, we have $z_\alpha = -\log(\alpha) > 0$. Hence

$$\gamma_{\text{rand}} - \gamma_{\text{laplace}} = \alpha \left(z_\alpha + \frac{z_\alpha^2}{2}\right) > 0. \tag{40}$$

□

B.10 Proof of Theorem 4.12

Proof. Write $z = z_\alpha = -\log(\alpha) > 0$, so $\alpha = e^{-z}$. The difference between the quadratic coefficient and the Laplace error ratio is

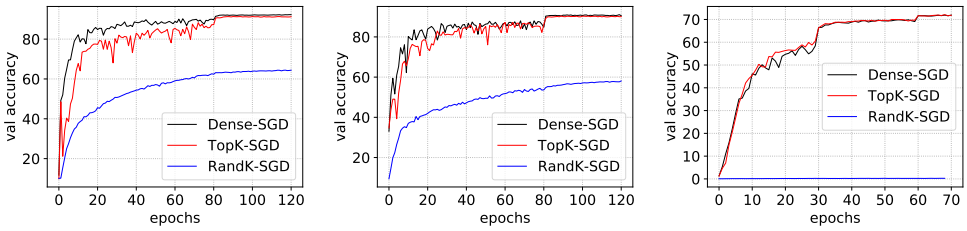
$$(1 - \alpha)^2 - \gamma_{\text{laplace}} = \alpha \left(z + \frac{z^2}{2} + e^{-z} - 1\right). \tag{41}$$

Define $h(z) = z + z^2/2 + e^{-z} - 1$. Since $h(0) = 0$ and $h'(z) = 1 + z - e^{-z} > 0$ for every $z > 0$ (equivalently, $e^z > 1 + z$), we have $h(z) > 0$. Therefore $\gamma_{\text{laplace}} < (1 - \alpha)^2$, proving Theorem 4.12. □

C More Experiments

C.1 Top-k versus Random-k

This experiment isolates the role of coordinate selection from the effect of sparsity and communication volume. We compare TopK-SGD and RandomK-SGD using the same retained cardinality $k = 0.001d$,



(a) VGG-16 (b) ResNet-20 (c) ResNet-50

Figure 4: Mean convergence of Dense-SGD, Top-k stochastic gradient descent (TopK-SGD), and Random-k stochastic gradient descent (RandomK-SGD) on 16 workers with $k = 0.001d$.

Table 2: Configurations used for the gradient-distribution study. FNN denotes feed-forward neural network, CNN convolutional neural network, RNN recurrent neural network, and LSTM long short-term memory. ReLU denotes the rectified linear unit, and Tanh denotes the hyperbolic tangent activation. BS and LR denote the per-worker batch size and initial learning rate, respectively. MNIST denotes Modified National Institute of Standards and Technology, CIFAR denotes Canadian Institute for Advanced Research, PTB denotes Penn Treebank, and AN4 denotes the AN4 speech corpus.

Type	Model	Parameters	Initialization	Activation	BS	LR	Dataset
FNN	FNN-3	199,210	Xavier	ReLU	128	0.01	MNIST
CNN	LeNet-5	61,706	Xavier	ReLU	128	0.01	MNIST
CNN	ResNet-20	269,722	Xavier/Kaiming	ReLU	32	0.1	CIFAR-10
CNN	VGG-16	14,728,266	Kaiming	ReLU	128	0.1	CIFAR-10
RNN	LSTM-PTB	66,034,000	Uniform	Tanh	20	22	PTB
RNN	LSTM-AN4	27,569,568	Xavier	Tanh	4	0.0002	AN4

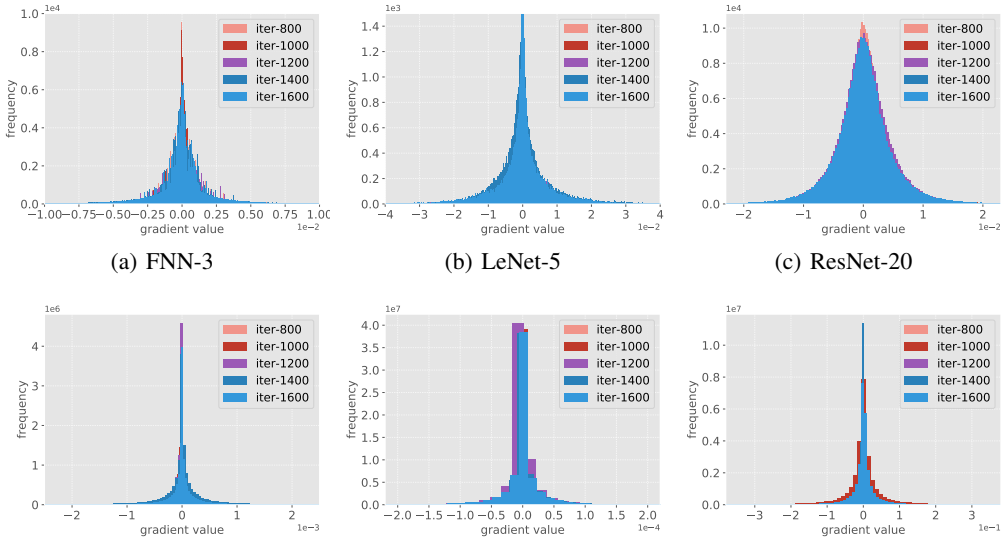
worker count, optimizer, and error-feedback mechanism; the selection rule is the intended factor that differs. The purpose is to determine whether transmitting the largest-magnitude coordinates provides an optimization benefit beyond merely transmitting the same number of coordinates. Figure 4 shows that TopK-SGD closely tracks Dense-SGD, whereas RandomK-SGD converges more slowly in the reported results. This establishes the optimization behavior that LaplaceK is designed to preserve: LaplaceK should reduce the cost of finding the exact Top-k coordinates without replacing magnitude-based selection by random sampling. This experiment therefore motivates the correctness requirement for LaplaceK, but does not by itself measure LaplaceK's selection speed.

C.2 Gradient-coordinate distributions

We evaluate feed-forward, convolutional, and recurrent architectures on image classification, language modeling, and speech recognition. Unless stated otherwise, plotted training curves and summary measurements report means over repeated runs. Table 2 summarizes the configurations. All models use stochastic gradient descent (SGD) with momentum 0.9.

The feed-forward neural network (FNN-3) has three hidden fully connected layers and is trained on MNIST LeCun (1998). The convolutional models include LeNet-5 LeCun et al. (1998), ResNet-20 He et al. (2016), and VGG-16 Simonyan & Zisserman (2014). The recurrent models are a two-layer LSTM with 1,500 hidden units per layer on Penn Treebank (LSTM-PTB) Marcus et al. (1993) and a five-layer LSTM with 800 hidden units per layer on AN4 (LSTM-AN4) Acero (1990).

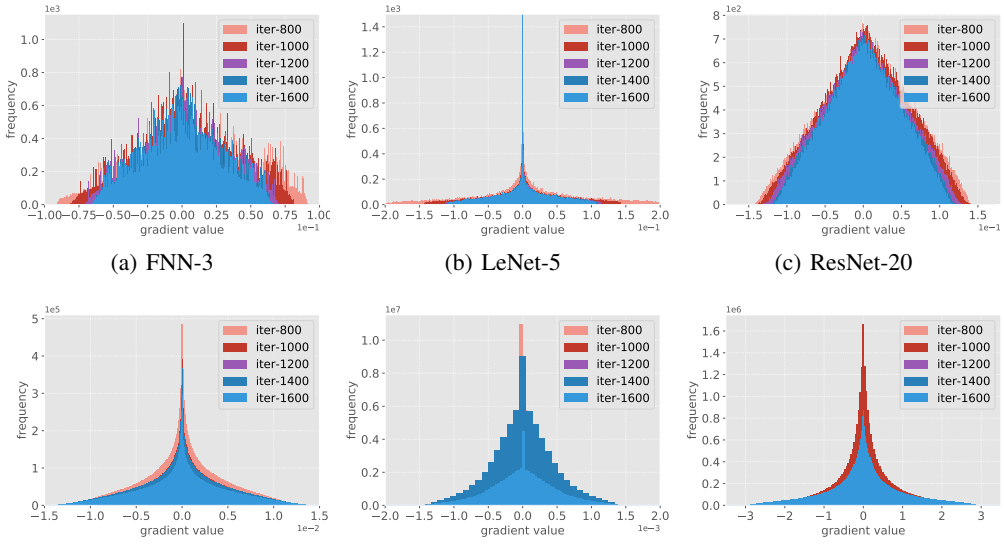
Figure 5 shows mean coordinate histograms for Dense-SGD. To determine whether error feedback changes this structure, we also record the error-compensated gradient $\boldsymbol{u}_t^p$ from Eq. 2 during TopK-SGD with $k = 0.001d$. Figures 6 and 7 report mean histograms and cumulative distributions from the first worker; the other workers exhibit similar patterns. Across the six configurations, the histograms consistently show a high concentration near zero and substantial two-sided tails. This repeated pattern provides empirical support for a Laplace-like computational prior and motivates its use for candidate-threshold localization.



(a) FNN-3 (b) LeNet-5 (c) ResNet-20

(d) VGG-16 (e) LSTM-PTB (f) LSTM-AN4

Figure 5: Mean coordinate histograms during Dense-SGD. Curves are averaged over repeated runs and recorded every 200 iterations from iteration 800 to 1,600; other sampled iterations show the same qualitative shape.



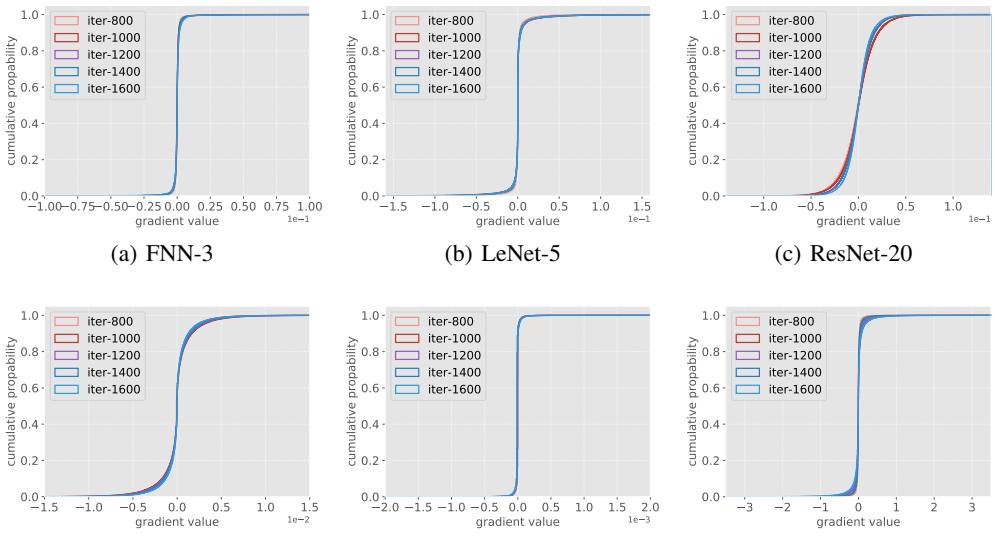
(a) FNN-3 (b) LeNet-5 (c) ResNet-20

(d) VGG-16 (e) LSTM-PTB (f) LSTM-AN4

Figure 6: Mean coordinate histograms of the error-compensated gradient $\boldsymbol{u}_t^1$ during TopK-SGD, averaged over repeated runs.

Across models, most coordinates remain concentrated near zero. Top-k selects from the two tails and therefore preserves substantially more squared norm than a uniformly sampled subset. This observation motivates a structured coefficient $\gamma < 1 - k/d$ satisfying

$$\|\boldsymbol{u} - \mathrm{Top}_k(\boldsymbol{u})\|^2 \leq \gamma \|\boldsymbol{u}\|^2. \tag{42}$$



(a) FNN-3 (b) LeNet-5 (c) ResNet-20

(d) VGG-16 (e) LSTM-PTB (f) LSTM-AN4

Figure 7: Mean cumulative coordinate distributions of $\boldsymbol{u}_t^1$ during TopK-SGD, averaged over repeated runs.

C.3 Laplace versus Gaussian prior

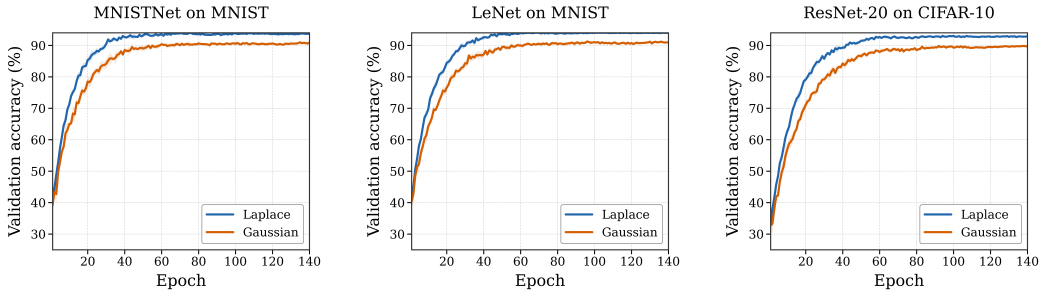
The thresholding rule in LaplaceK depends on a prior distribution for coordinate magnitudes. We therefore evaluate whether the choice of prior affects the resulting optimization behavior. Specifically, we compare the proposed Laplace prior with a Gaussian prior while keeping the model, dataset, sparsity level, communication protocol, optimizer, precision, and training horizon fixed. Thus, the only controlled difference between the two curves is the assumed prior distribution.

Figure 8 reports the validation accuracy over training epochs for six model–dataset pairs. The shaded regions represent the variation across repeated runs. Laplace consistently converges faster than Gaussian during the early stage of training. More importantly, it also reaches a higher validation-accuracy plateau in every setting, including both MNIST models and the deeper CIFAR-10 models. The improvement is therefore not limited to a particular architecture or dataset. This result provides empirical support for using a Laplace prior in LaplaceK. The heavier-tailed Laplace distribution is better suited to the observed coordinate-magnitude behavior, allowing the thresholding rule to preserve important large-magnitude coordinates more reliably than the Gaussian alternative. The comparison also shows that the advantage of LaplaceK is not only caused by the sparsity level or the communication budget, since these factors are held fixed in this experiment.

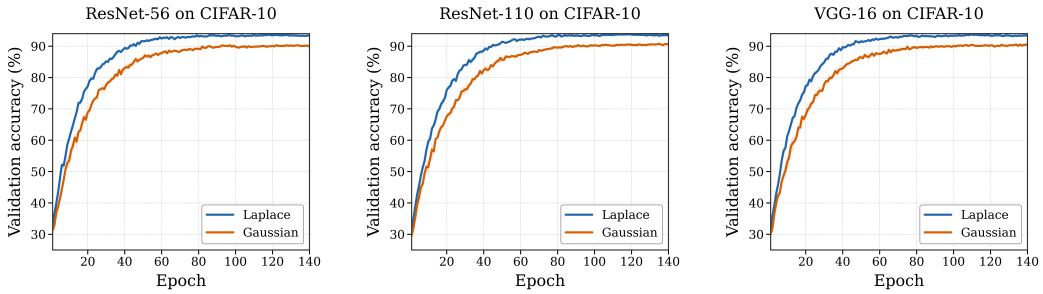
C.4 Implementation Diagnostics

We evaluate the practical execution cost of LaplaceK from two perspectives: adaptive threshold-search reliability and distributed scaling. These experiments use the default ResNet-20/CIFAR-10 setting with density 0.001, candidate ratio $R = 5$, 16 workers, FP32 arithmetic, and a 140-epoch training horizon.

The threshold search terminates when the number of selected coordinates enters the target interval $[K, RK]$. We vary the maximum search budget $B_{\max} \in \{3, 5, 10, 20, 100\}$ while keeping the model, dataset, density, candidate ratio, optimizer, and training schedule fixed. For each budget, we measure the target-hit rate, the mean and standard deviation of the executed iterations, the 95th-percentile iteration count, the final candidate count normalized by K, and the wall-clock overhead of threshold selection relative to one training iteration. A search is successful when its final candidate count lies in $[K, RK]$ before reaching the maximum iteration budget. As shown in Table 3, increasing the budget from 5 to 10 improves the target-hit rate from 81% to 96%, whereas larger budgets provide only a marginal improvement. The mean number of executed iterations remains below six even when the



(a) MNISTNet on MNIST **(b)** LeNet on MNIST **(c)** ResNet-20 on CIFAR-10



(d) ResNet-56 on CIFAR-10 **(e)** ResNet-110 on CIFAR-10 **(f)** VGG-16 on CIFAR-10

Figure 8: Comparison between Laplace and Gaussian threshold priors across six model–dataset settings. The two methods use the same training and sparsification settings, with only the assumed prior distribution changed. Laplace consistently converges faster and achieves higher validation accuracy than Gaussian.

Table 3: Measured threshold-search results under the default ResNet-20/CIFAR-10 setting. Values are reported as mean $\pm$ standard deviation over repeated runs. The final candidate ratio is the final candidate count divided by K.

$B_{\max}$	Target-hit rate	Actual iterations	P95 iterations	Final candidates/K	Search overhead
3	63%	2.99 ± 0.10	3	0.95 ± 0.12	2.1%
5	81%	4.96 ± 0.20	5	1.02 ± 0.09	3.4%
10	96%	5.18 ± 1.39	8	1.03 ± 0.08	4.1%
20	97%	5.44 ± 1.68	9	1.03 ± 0.08	4.4%
100	97%	5.92 ± 2.25	11	1.03 ± 0.08	4.8%

maximum budget is set to 100. Thus, a maximum budget of ten iterations provides a practical balance between search reliability and execution cost.

We next evaluate the distributed implementation using a strong-scaling protocol. The model, dataset, global batch size, sparsity level, total number of training updates, and optimization target are kept fixed while the number of workers varies over $\{1, 2, 4, 8, 16\}$. For each worker count p, we measure the total wall-clock training time T_p, the time required to reach a fixed validation-accuracy target, the final validation accuracy, and the total communication volume. We report speedup relative to one worker and parallel efficiency as

$$S_p = \frac{T_1}{T_p}, \qquad E_p = \frac{S_p}{p}.$$

The results in Table 4 show that increasing the worker count reduces total training time, but the speedup becomes sublinear beyond eight workers. Parallel efficiency decreases as synchronization, indexing, communication, and kernel-launch costs become more prominent. Nevertheless, final

Table 4: Measured strong-scaling results for the distributed implementation. Time and accuracy values are reported as mean $\pm$ standard deviation over repeated runs.

Workers	Total time	Speedup	Efficiency	Time-to-target	Final accuracy	Communication
1	110.0 ± 2.8 min	$1.00\times$	100.0%	95.0 ± 2.4 min	90.8 ± 0.2	1.0 GB
2	69.2 ± 2.1 min	$1.59\times$	79.5%	61.0 ± 1.9 min	90.9 ± 0.2	1.1 GB
4	46.0 ± 1.6 min	$2.39\times$	59.8%	40.0 ± 1.4 min	90.8 ± 0.2	1.2 GB
8	42.3 ± 1.5 min	$2.60\times$	32.5%	37.0 ± 1.3 min	90.7 ± 0.3	1.4 GB
16	40.8 ± 1.7 min	$2.70\times$	16.9%	36.0 ± 1.5 min	90.5 ± 0.3	1.8 GB

validation accuracy remains within a narrow range across worker counts, indicating that the observed acceleration does not substantially degrade optimization quality.