LBI: PARALLEL SCAN BACKPROPAGATION VIA LATENT BOUNDED INTERFACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Backpropagation is sequential across depth, an $O(K)$-deep dependency chain that bottlenecks parallel training. Parallel-scan formulations reduce the depth to $O(\log K)$ but compose $d \times d$ Jacobians at $O(d^3)$ per combine. We introduce Latent Bounded Interfaces (LBI), which restrict inter-region communication to an interface $m_k \in \mathbb{R}^r$ with $r \ll d$, so the adjoint recursion becomes a suffix scan over $r \times r$ Jacobians with the gradient exact under the bounded-interface model. The same restriction bounds what crosses a device boundary: per-sequence states and $r \times r$ Jacobians in place of per-token activations, so a region-parallel training step's critical path is kilobytes per link regardless of width, and its one activation-sized transfer is a gradient that can be accumulated. Implemented as fused forward-mode kernels on four H100s, the step runs $14\times$ faster than pipeline parallelism at 1 Gbit/s and $30\times$ faster at 100 Mbit/s of link bandwidth, which extends exact-gradient training to multi-region and decentralized links. On language modeling, bounded-interface models train across Mamba-3, Transformer, and Hybrid backends at small interface ranks.

1 INTRODUCTION

Backpropagation in deep neural networks is inherently sequential across layers: the adjoint at layer k depends on the adjoint at layer $k{+}1$, which depends on layer $k{+}2$ and so on. The chain has depth $O(K)$ in the number of layers or regions K. Exact scan-based approaches, such as BPPSA (Wang et al., 2020), reduce the $O(K)$ dependency chain to $O(\log K)$, but are computationally prohibitive due to the $O(d^3)$ cost of composing $d \times d$ Jacobians (where $d = BLD$ encompasses the full batch, sequence, and feature dimensions). Iterative (Fagnou et al., 2025; Lim et al., 2023; Gonzalez et al., 2024; Günther et al., 2020), decoupled (Huang et al., 2025; Jaderberg et al., 2017), and Jacobian-Free (Fung et al., 2022) approaches sacrifice gradient exactness for throughput. Pipeline parallelism (Gomez et al., 2022; Li & Hoefler, 2021; Huang et al., 2019) uses micro-batch scheduling to improve hardware utilization but leaves the $O(K)$ sequential depth intact and moves activation-sized payloads across every stage boundary per microbatch, which confines it to high-bandwidth interconnects or to lossy compression of those payloads (Wang et al., 2022; Ramasinghe et al., 2026).

We observe that the intractability of scan-based backpropagation is not inherent to the scan formulation itself, but to the size of the objects being scanned. Restricting inter-region communication to a low-dimensional *interface state* $m_k \in \mathbb{R}^r$ reduces the adjoint recursion to a suffix scan over $r \times r$ Jacobians, preserving the $O(\log K)$-depth associative structure identified by BPPSA while cutting per-combine cost from $O(d^3)$ to $O(r^3)$ with $r \ll d$. All remaining gradient computation (parameter gradients and intra-region adjoints) becomes embarrassingly parallel across regions once the interface adjoints are available. The same restriction bounds what crosses a device boundary: per-sequence states and $r \times r$ Jacobians in place of per-token activations, so a region-parallel step's critical path is kilobytes per link regardless of width, and its one activation-sized transfer is a gradient, which can be accumulated across steps. Training that keeps the gradient exact can then run over links a pipeline cannot use without compressing what it sends.

Inter-component communication in neural networks admits learned low-dimensional compression when it is native to the model and learned end-to-end, as Multi-Head Latent Attention (MLA) shows for attention (DeepSeek-AI et al., 2024). We apply the same principle across depth: the model learns a compact interface carrying only what downstream computation needs. The resulting *bounded-*

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.

interface model is at once a modeling choice, a structural information bottleneck, and the enabler that reduces the scan operator from $d \times d$ to $r \times r$. We make the following contributions:

1. **Tractable Scan Backpropagation via Bounded Interfaces.** We restrict inter-region communication to an r-dimensional interface and prove that the resulting adjoint recursion admits an exact suffix scan over $r \times r$ Jacobians, reducing per-combine cost from $O(d^3)$ to $O(r^3)$ with $O(\log K)$ depth.
2. **Three-Phase Decomposition with Region Independence.** We show that bounded-interface backpropagation decomposes into three phases: embarrassingly parallel Jacobian construction, a lightweight suffix scan, and embarrassingly parallel region-local backward, where the scan is the sole sequential bottleneck at $O(r^3 \log K)$ span.
3. **Exactness without Approximation.** We prove that the scan-based backward recovers the gradients of standard reverse-mode backpropagation under the bounded-interface model, and verify it numerically to machine precision in float32. Unlike prior approaches, it introduces no approximation, staleness, or auxiliary objective.
4. **Measured Realization.** We implement the construction as fused forward-mode kernels and a region-parallel training step and measure it on four H100s. The step's critical path carries kilobytes per link against a pipeline parallel's 67 MB, so on a shaped link it runs $14\times$ faster than a tuned pipeline at 1 Gbit/s and $30\times$ faster at 100 Mbit/s, which extends synchronous exact-gradient training to multi-region and decentralized links. On NVLink, it runs at $0.88\times$ the pipeline's speed. On language modeling from 47M to 321M backend parameters, bounded interfaces cost 0.33–0.67 CE, a gap set by region granularity. The code is available at `https://anonymous.4open.science/r/latent-bounded-interfaces-33ED/`.

2 Scan Formulation

We formalize the notion of a bounded interface as a low-dimensional vertex separator in the computational graph through which all inter-region dependencies must pass, and show that this structure implies a factorization of both forward computation and reverse-mode adjoints. Proofs of the results of Sections 2 and 3 are in Appendix A.

2.1 Graph and Interface Factorization

Definition 2.1 (Region-Partitioned Computational Graph)**.** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote the directed acyclic computational graph of a model, where $\mathcal{V}$ are the graph nodes and $\mathcal{E}$ are directed edges representing data dependencies. A *region partition* of $\mathcal{G}$ is a disjoint decomposition, $\mathcal{V} = \bigsqcup_{k=0}^{K-1} \mathcal{R}_k$ that is *topologically ordered*: for every edge $(u \to v) \in \mathcal{E}$ with $u \in \mathcal{R}_k$ and $v \in \mathcal{R}_j$, we have $k \leq j$. This ensures that inter-region dependencies flow in a single forward direction and permits arbitrary region boundaries consistent with the DAG, including partitions that do not align with architectural layers. For each boundary between regions $\mathcal{R}_{k-1}$ and $\mathcal{R}_k$, we associate an *interface state,* $m_k \in \mathbb{R}^{r_k}$, where r_k denotes the interface dimension, which controls the amount of information communicated across region boundaries.

Definition 2.2 (Interface Sufficiency)**.** An interface state m_k is *sufficient* at boundary k if for every node $v \in \mathcal{R}_{\geq k}$, there exists a function $\tilde{f}_v$ such that $v = \tilde{f}_v(m_k; \theta_{\geq k})$. That is, all dependence of v on variables in $\mathcal{R}_{<k}$ factors exclusively through m_k.

Sufficiency holds whenever m_k forms a *vertex separator* between $\mathcal{R}_{<k}$ and $\mathcal{R}_{\geq k}$ in the computational graph, that is, every directed path from $\mathcal{R}_{<k}$ to $\mathcal{R}_{\geq k}$ passes through nodes whose values determine m_k. A formal proof that a vertex separator implies sufficiency (Definition 2.2) is given in Appendix B (Lemma B.2). In our construction (Section 3), sufficiency holds by design. The architecture routes all inter-region communication exclusively through the interface states, with no bypass paths.

Lemma 2.3 (Adjoint Factorization Through Interfaces)**.** *If m_k is sufficient at boundary k, then for any loss $\mathcal{L}$ computed from variables in $R_{\geq k}$,*

$$\frac{\partial \mathcal{L}}{\partial \mathcal{R}_{<k}} = \left(\frac{\partial m_k}{\partial \mathcal{R}_{<k}}\right)^{\top} \frac{\partial \mathcal{L}}{\partial m_k}. \tag{1}$$

That is, all reverse-mode adjoints from $\mathcal{R}_{\geq k}$ to $\mathcal{R}_{<k}$ factor exclusively through the interface m_k.

Definition 2.4 (Bounded-Interface Model)**.** A region-partitioned computational graph is said to satisfy the *bounded-interface property* if, for every boundary k, there exists a sufficient interface state m_k with $\dim(m_k) \leq r_k$ for some prescribed interface budget r_k independent of the total number of internal variables in the adjacent regions.

Remark 2.5 (Imposed vs. Discovered Interfaces). By Menger's theorem (Böhme et al., 2001), the minimum vertex separator between $\mathcal{R}_{<k}$ and $\mathcal{R}_{\geq k}$ in a typical neural network computational graph has size d_k (the full hidden state) since each scalar component admits an independent path across the boundary under non-degeneracy (Appendix B). The bounded-interface property is therefore an *architectural imposition*, not a consequence of the model's existing structure. We choose to communicate $r_k \ll d_k$ scalars where exact sufficiency would formally require d_k. The architectural and empirical factors that make this compression viable are addressed in Sections 3 and 4.

Corollary 2.6 (Sequential Interface Representation)**.** *For a bounded-interface model, all inter-region dependence factors through the interface chain $m_0 \to m_1 \to \cdots \to m_K$. In particular, inter-region forward computation is completely characterized by the transition maps,*

$$m_{k+1} = R_k(m_k; \theta_k), \tag{2}$$

where R_k denotes the interface transition map associated with region $\mathcal{R}_k$, while all remaining computation is local to its corresponding region with local parameters θ_k. By Lemma 2.3, reverse-mode transport across regions must therefore also factor through this chain.

Corollary 2.6 establishes that both forward and backward computation across regions reduce to operations on the interface chain $m_0 \to m_1 \to \cdots \to m_K$ rather than on the full hidden state. Since each interface transition $m_k \to m_{k+1}$ is a differentiable map between fixed-dimensional spaces, the inter-region adjoint recursion $\bar{m}_k = J_k^\top \bar{m}_{k+1}$ composes linear operators of size $r \times r$, an associative structure that admits a parallel scan formulation.

2.2 Scan Parallel Backpropagation

Backpropagation over bounded interface models follows directly from Lemma 2.3 and Corollary 2.6, admitting a parallel scan formulation and span advantage over standard sequential backpropagation. For simplicity in analysis and instantiation (Sections 2.2–4), we assume a uniform interface width $r_k = r$. The variable-width case is analogous. We refer to the propagation of adjoints across region boundaries via Jacobian transposes as *adjoint transport*. Concretely, for a region transition $m_{k+1} = f_k(m_k)$, $\bar{m}_k = (\partial m_{k+1}/\partial m_k)^\top \bar{m}_{k+1}$. The interface chain (Corollary 2.6) is seeded by an initial state m_0 and terminates at m_K. We assume that the loss $\mathcal{L}$ depends on all preceding regions exclusively through m_K, so that $\bar{m}_K = \partial \mathcal{L}/\partial m_K$ is well-defined and nonzero.

Theorem 2.7 (Bounded-Interface Backpropagation)**.** *Let $\mathcal{G}$ be a region-partitioned computational graph satisfying the bounded-interface property, with region transitions as in Eq. 2, and let $\mathcal{L}$ be a scalar loss depending only on variables in the final region $\mathcal{R}_{K-1}$.*

Define the interface adjoint $\bar{m}_k := \partial \mathcal{L}/\partial m_k \in \mathbb{R}^r$. Then, for every $k = 0, \ldots, K-1$, $\bar{m}_k = \left(\frac{\partial m_{k+1}}{\partial m_k}\right)^\top \bar{m}_{k+1}$, and the local parameter gradient factorizes as $\nabla_{\theta_k} \mathcal{L} = \left(\frac{\partial m_{k+1}}{\partial \theta_k}\right)^\top \bar{m}_{k+1}$. Moreover, writing $J_k := \partial m_{k+1}/\partial m_k \in \mathbb{R}^{r \times r}$, where the operator J_k (the Jacobian) is the differential of the region transition map evaluated at the forward state m_k, the inter-region adjoint admits the exact product form,

$$\bar{m}_k = \left(\prod_{j=k}^{K-1} J_j^\top \right) \bar{m}_K, \quad k = 0, \ldots, K-1, \tag{3}$$

with the product ordered as $J_k^\top J_{k+1}^\top \cdots J_{K-1}^\top$, and the empty product ($k = K$) is the identity.

The product form in Eq. 3 shows that inter-region adjoint transport reduces to the composition of linear operators $\{J_k^\top\}$. Since matrix multiplication is associative, this composition admits a parallel scan formulation.

Proposition 2.8 (Scan Formulation of Backpropagation)**.** *The inter-region adjoints $\{\bar{m}_k\}_{k=0}^K$ admit an exact suffix scan formulation over the sequence $\{J_k^\top\}_{k=0}^{K-1}$. In particular, defining $P_k := \prod_{j=k}^{K-1} J_j^\top$, we have $\bar{m}_k = P_k \bar{m}_K$, where the products $\{P_k\}$ can be computed in parallel using any associative scan algorithm.*

This replaces the sequential dependency chain of reverse-mode backpropagation with a logarithmic-depth parallel scan over fixed-size operators.

Corollary 2.9 (Region-Local Backward Independence)**.** *Following Theorem 2.7, once the interface adjoints $\{\bar{m}_k\}$ are available, all region-local backward computations are independent across regions.*

Proposition 2.8 and Corollary 2.9 establish the structural decomposition. Crucially, this structure holds for any interface dimension, including $r_k = d_k$ (recovering BPPSA's full-rank scan (Wang et al., 2020)). What separates the bounded-interface regime from the full-rank regime is not the existence of an associative decomposition, but the cost of each scan operation. We address this in Section 3.

3 Model Realization

3.1 Bounded-Interface Architecture

Instantiation Template. We instantiate a bounded-interface model by associating each region k with a local computation over activations x_k and a low-dimensional interface state $m_k \in \mathbb{R}^r$. At each region k, the forward computation is defined as

$$x_k^{\text{in}} = x_{\text{embed}} + \text{Dec}_k(m_k), \quad x_k^{\text{out}} = \Phi_k(x_k^{\text{in}}), \tag{4}$$

where Φ_k denotes the region transformation map (e.g. attention, SSM, MLP), and Dec_k injects the interface state into the region. The interface state is then updated by

$$m_{k+1} = R_k(m_k) = \text{LN}(m_k + \alpha_k \,\text{Enc}_k(\text{pool}(\Phi_k(x_{\text{embed}} + \text{Dec}_k(m_k))))), \tag{5}$$

where Enc_k projects region outputs into the interface space, $\text{pool}(\cdot)$ aggregates region outputs, α_k is a learned per-region scaling parameter, and LN denotes layer normalization (Ba et al., 2016). The initial interface state $m_0 \in \mathbb{R}^r$ is derived from the input tokens, and the terminal state m_K serves as input to the terminal block / loss head. Thus, the construction in Eq. 5 induces a region-partitioned computational graph in which all inter-region dependencies are mediated by the interface states $\{m_k\}_{k=0}^K$, thereby realizing the abstract region transition map $R_k(m_k; \theta_k)$ of Corollary 2.6 in a concrete architectural form. Notice that when conditioned on m_{k+1}, no variable in region $\mathcal{R}_{\geq k+1}$ depends on variables in $\mathcal{R}_{\leq k}$, so the bounded-interface property holds.

Interface Design and Coupling. A key design constraint in bounded-interface models is that all dynamic inter-region dependencies must be mediated through m_k, with no activation bypass paths from x_k^{out} to subsequent regions. Instead, we inject a shared static signal x_{embed} at each region (Eq. 4), providing every region with a common representational canvas. The interface state m_k then needs to encode only the refinement from preceding regions rather than the full hidden representation, which explains why the interface rank can be small relative to the model dimension. The canvas does not violate the bounded-interface property (Definition 2.4): x_{embed} depends only on the input tokens, not on any region's computation, and therefore does not constitute an inter-region communication channel. Our implementation uses 2-layer MLPs for the encoder and decoder, which introduces $\Theta(D_k^2)$ parameters per encoder/decoder pair, and is comparable in order to the region transformation Φ_k itself. Alternative encoder/decoder realizations are discussed in Section 6.

3.2 Computational Cost

The cost of bounded-interface backpropagation decomposes into three components: Jacobian construction, scan composition, and region-local backward. We state the structural work–span (Blelloch, 1996) result first, then derive each term.

Proposition 3.1 (Work–Span Decomposition)**.** *The total work of bounded-interface backpropagation decomposes as,*

$$W_{\text{bi}} = \sum_{k=0}^{K-1} W_k^J + \Theta\left(Kr^3\right) + \sum_{k=0}^{K-1} W_k^{\text{local}}, \tag{6}$$

and the total span satisfies

$$T_{\text{bi}} = \max_k W_k^J + \Theta(r^3 \log K) + \max_k W_k^{\text{local}} \tag{7}$$

where the three terms correspond to (i) Jacobian construction across all regions (Eq. 8), (ii) scan composition over interface Jacobians (Proposition 2.8), and (iii) region-local backward computation (Corollary 2.9).

Jacobian Construction. Each $J_k := \partial m_{k+1}/\partial m_k \in \mathbb{R}^{r\times r}$ is constructed by propagating r_k perturbation directions through the linearized region R_k via automatic differentiation (AD) (Baydin et al., 2017). Appendix D details the construction procedure and its implementation. Let F_k and Q_k denote the total forward FLOPs and memory traffic for region k. Jacobian construction augments the forward computation with a basis dimension of size r, yielding total work

$$W_k^J = \mathrm{FLOPs}(J_k) = \Theta(rF_k). \tag{8}$$

Under activation reuse across perturbation directions, memory traffic is nearly unchanged, an r-fold increase over the forward pass in arithmetic intensity. The materialized Jacobian itself requires only r^2 scalars per region which is negligible relative to activation storage. On the roofline time model (Williams et al., 2009) this reuse consolidates to an absorption factor

$$g(r) := \max(1, rI_{\mathrm{fwd}}/I_{\mathrm{ridge}}), \tag{9}$$

the construction's idealized cost in forward-pass equivalents, where I_{fwd} is the region forward's arithmetic intensity and I_{ridge} is the device's ridge point. On a memory-bound region the tangent arithmetic occupies compute the forward leaves idle, and $g(16) = 1$ on the SSM region. We write c for the construction's measured cost in the same units, the wall-clock time to construct J_k divided by the region's forward time. $g(r) \le c \le r$ brackets it between perfect reuse and one forward pass per direction, and the span comparison of Proposition 3.1 places the wall-clock crossover against sequential backpropagation at $K \gtrsim c{+}1$ (Appendix C). Section 4.2 measures c. The scan is negligible beside the construction (Appendix C), and each region's local backward costs $W_k^{\mathrm{local}} = \Theta(F_k)$, as in standard reverse mode seeded with $\bar{m}_{k+1}$ (Corollary 2.9), so the exchange is $(r{+}1)\times$ the arithmetic for $O(\log K)$ scan depth. Algorithm 1 states the three-phase procedure (with the schedule that overlaps its phases in Appendix D).

4 Experiments

4.1 Training Quality

Training Setup. We evaluate bounded-interface backpropagation across three backends (Mamba-3 SISO (Lahoti et al., 2026), Transformer (Vaswani et al., 2017), and a hybrid interleaving three Mamba-3 blocks with one Transformer block) at 120–157M backbone parameters, comparing dense and LBI variants at identical depth, width, and training budget. All models train on 2.95B tokens of FineWeb-Edu (Penedo et al., 2024) with the 32k LLaMA tokenizer (Touvron et al., 2023) at context 1024 (batch 32 for 90k steps, the Transformer token-matched at batch 16 for 180k). The learning rate is tuned per configuration on the first 15k steps of the schedule, and every number is the final checkpoint's cross-entropy (CE) on a fixed 4.2M-token held-out sample. Regions are two consecutive blocks and the interface is the MLP realization of Section 3.1 at $r = 16$. Intra-region parameters are held fixed rather than matched on totals. Further protocol details are in Appendix E.

Backend	Scale	K	Dense	LBI	Gap
Mamba-3	18L/1024d (120M + 20.3M)	9	2.669 ± 0.001	3.212 ± 0.002	0.54
Transformer	10L/1024d (157M + 11.7M)	5	2.667 ± 0.001	3.001 ± 0.005	0.33
Hybrid	16L/1024d (143M + 18.1M)	8	2.630 ± 0.010	3.077 ± 0.001	0.45

Table 1: LBI models against dense models of identical depth and width, final-checkpoint validation CE, mean $\pm$ standard deviation over three seeds. K is the number of regions. The scale column gives the backend parameters both models share, and the interface parameters only the LBI model carries after the $+$.

Bounded-interface backpropagation recovers the gradients of standard reverse-mode AD on the same bounded-interface model to machine precision in float32, and in bfloat16 to a discrepancy that grows with depth from rounding along the chain (Table 8, Appendix E). Table 1 reports the gap: 0.33–0.54

CE at this budget. On Mamba-3 under the same protocol the gap moves within 0.17 CE across a $6\times$ range of scale, from 0.51 at 54M to 0.63–0.67 at 321M (Table 11). It falls as regions coarsen, from 0.53 at two-block regions to 0.16 at nine-block regions on the stack of Table 1 at batch 16 (Table 12), and does not change with the interface rank over the ranges measured. Validation CE is flat from $r = 8$ to $r = 64$ on a 15k-step truncation of the schedule on all three backends, and at the full budget the Mamba-3 gap is 0.54 CE at $r = 8, 16, 32$ (three seeds each, the Hybrid at $r = 8$ trains to the same gap, Table 9). A dense Mamba-3 matched on total parameters instead of depth trains to 2.643, which widens the gap to 0.57, so the gap is not a capacity effect (Appendix E).

4.2 Region Parallelism on Hardware

We implement Algorithm 1 as fused forward-mode kernels and a region-parallel step in which device k owns region k, the forward pass is a chain of $[B, r]$ messages, and the Jacobian construction of earlier regions overlaps the chain fill (the forward pass of the message chain through the later regions).

Hardware Setup. The hardware measurements use a smaller model than Section 4.1, an 8-block stack of four two-layer regions ($K = 4$, one region per device and one pipeline stage each) at width 1024, context 2048, batch 8, on 4 H100s. The baselines are the same stack trained by sequential backpropagation on one device, and by pipeline parallelism with torch's GPipe and 1F1B schedules (Huang et al., 2019; Narayanan et al., 2019), shown at its fastest configuration, GPipe with $M = 4$ microbatches (the tuned pipeline, PP), and at its lowest-memory configuration, 1F1B with $M = 8$. The other microbatch configurations are in Appendix D. We write RP for the region-parallel step with the canvas gradient reduced every step and RP-N for the step with an N-step window. For communication we count bytes on the critical path (Blelloch, 1996), which is what a device must receive before it can proceed. A pipeline step moves $2(K-1)$ full-batch activation tensors, all on the critical path since a stage cannot start a microbatch until its input has arrived (Huang et al., 2019). Two instruments measure step time: an *emulated link*, the real step on NVLink with $\mathrm{RTT} + \mathrm{bytes}/\mathrm{bandwidth}$ charged to the sender at each transfer, and a *shaped link*, NCCL over TCP on a loopback link rate-limited with Linux traffic control and shared by all 4 devices, as in Wang et al. (2022); Ryabinin et al. (2023). Both run at 5 ms round-trip time (RTT), the same-region cloud latency of Yuan et al. (2022); Ryabinin et al. (2023), on Mamba-3 at $r = 8$, the smallest rank at which quality is flat (Section 4.1). Both agree within 5% below 1 Gbit/s with the latency–bandwidth bound of the α–β model (Hockney, 1994), $T \geq \max(T_0,\ \mathrm{bytes}_{\mathrm{link}}/\mathrm{BW} + n_{\mathrm{serial}}\,\mathrm{RTT})$ with T_0 the network-free step (Appendix D).

	wire MB/step	critical-path bytes/link	bandwidth to 95%	100 Mbit/s emul.	100 Mbit/s shaped	1 Gbit/s emul.	1 Gbit/s shaped	10 Gbit/s emul.	10 Gbit/s shaped
PP (tuned pipeline)	201	67.1 MB	77 G	5477	16152	643	1603	154	158
RP (every step)	201	6.4 KB	20 G	4092	16143	470	1604	105	232
RP-8	25	6.4 KB	7.7 G	562	2065	109	253	63	86
RP-32	6	6.4 KB	3.1 G	186	543	73	113	61	72

Table 2: Communication on four H100s (Mamba-3, $r = 8$, the shape of Table 3). Step time in ms at RTT 5 ms on the emulated and shaped links, medians of three runs (every-step: one run). Bandwidth to 95%: the link rate at which the step is within 5% of its network-free step at 100 Gbit/s on the emulated link at RTT 0.1 ms.

Communication over Weak Interconnects. A region-parallel step sends a $[B, r]$ state across each boundary, the $[B, r]$ loss adjoint back, and each device's share of the all-reduce of the K per-sequence $r \times r$ Jacobians, 6.4 KB per link at $r = 8$ and independent of width ($\Theta(KBr^2 + Br)$). The only activation-sized transfer is the gradient with respect to the shared canvas (*canvas gradient*), and nothing downstream waits for it. In wire bytes (ring all-reduce traffic (Patarasuk & Yuan, 2009)) RP moves the same 201 MB per step as PP. The difference is 67 MB per link on the critical path against 6.4 KB (Table 2). Because the canvas gradient is a gradient, an N-step window (RP-N) sums it over the window and reduces it once, as in low-communication training (Douillard et al., 2023), dividing the wire bytes by the window length with the canvas learning rate scaled for its larger effective batch. At the full budget RP-32 trains to 3.214 against 3.212 ± 0.002 CE (Table 10). PP's bytes are activations, sent once per microbatch, and cannot be accumulated. RP-32 runs $30\times$ / $14\times$ / $2.2\times$ faster than PP at 100 Mbit/s / 1 Gbit/s / 10 Gbit/s on the shaped link (step-time ratio at equal

work per step, Table 2), $7.8\times$ / $3.3\times$ faster than an emulated 4-bit-compressed PP at 100 Mbit/s / 1 Gbit/s (Wang et al., 2022; Ramasinghe et al., 2026), and $7.8\times$ at width 2048 and $25\times$ / $6.3\times$ on the Hybrid (Appendix D). The RP step's latency exposure is four serial round trips per step (three chain messages and the adjoint broadcast, the Jacobian all-reduce overlaps the chain and the shared scalar's reduce joins the window collective, Appendix D), 4.1–4.3 ms per millisecond of RTT on the shaped link, against PP's 11–20 ms at 10 Gbit/s, one round trip per stage boundary per microbatch. So at 50 ms RP-32 remains $5.5\times$ / $4.1\times$ faster at 1 / 10 Gbit/s (Table 7). The bandwidth at which each step is within 5% of of its latency-only floor, its step at 100 Gbit/s on the same link (the utilization statistic of (Douillard et al., 2025)) is about 77 Gbit/s for PP, 7.7 Gbit/s for RP-8, and 3.1 Gbit/s for RP-32. At the 1 Gbit/s of cross-region cloud links (0.3–2 Gbit/s (Yuan et al., 2022)) RP-32 runs within $2\times$ of its network-free step and PP about $30\times$ (Table 2).

Wall-Clock and Memory. Against PP the RP step at $r = 8$ runs at $0.88\times$ its speed on Mamba-3 (59.4 ms against 52.0 ms), $0.59\times$ on the Hybrid and $0.42\times$ on the Transformer, and at $r = 16$ at $0.63/0.43/0.32\times$ (Table 3). The step falls as the interface narrows because the Jacobian construction is the dominant term. The Transformer column is farthest because its regions are compute-bound, so its construction costs $0.42r$ forward passes with no primal work to reuse (Appendix C). No four-device scheme beats the sequential reference at this scale (36.4 ms on Mamba-3): $K = 4$ lies below the crossover $K \gtrsim c + 1$ of Section 3.2, which the construction constant c measured below places near $K = 16, 9$ at $r = 16, 8$ (Appendix C). On Mamba-3 at $r = 16$, the state a device holds across the step is 1.7 GB (replicated weights, its region's forward cache, and the canvas). The rest of the peak is transient, in the Jacobian construction and the region backward. Both scale with the number of sequences and tangent directions processed at once, so both can be split: driving the r tangent directions through the region in chunks of 8 / 4 / 2 brings the peak to 6.3 / 4.5 / 4.4 GB for 4 / 12 / 32% more wall clock with the same gradients in exact arithmetic, and microbatching the step as PP does divides both transients again, to 2.5 GB in 135 ms with four microbatches and 2.2 GB in 223 ms with eight (Table 5, Appendix D). The latter is under the PP's 2.3 GB at four times its step, and above the lowest-memory pipeline's 1.2 GB (106 ms). The Table 3 rows, one microbatch with all r directions in one pass, sit above both pipelines because that construction transient dominates the footprint.

Construction and Profiling. On the training configuration of Table 1 (18-layer Mamba-3, $r = 16$, batch 32, context 1024) measured on 1 H100, the Jacobian construction costs $c = 14.5$ forward passes (905 ms against a 62.3 ms forward), near the no-reuse end $c = r$ of Section 3.2, about one forward pass per tangent direction. A kernel-level breakdown attributes 61% to the two kernels of the chunked SSM scan (the intra-chunk scan (31%) and the inter-chunk recurrence that advances the SSM state serially from chunk to chunk while carrying all r tangent directions (30%)), and 39% to the interface projections, norms, and pointwise work. The forward-mode construction used throughout this section takes 1200 ms per step at 43.5 GB peak memory, against 2056 ms and 45.3 GB for a reverse-mode construction, and 202 ms and 45.2 GB for end-to-end autograd (three interchangeable engines outlined in Appendix D). *Scaling.* The constant depends on the region kernel and the token count, not on sequence shape: construction is linear in r, 481 / 905 / 1800 ms at $r = 8/16/32$, or 56–60 ms per tangent direction, the no-reuse end of Section 3.2. At a fixed token count the sequence shape does not affect construction, halving the batch at doubled context reproduces it to 1.1% (Appendix C) *Hardware counters.* the inter-chunk recurrence kernel runs at 12.5% achieved occupancy, its cap at the kernel's register and shared-memory footprint, with streaming-multiprocessor throughput at 32% and no memory unit above half its ceiling. It is latency-bound (not compute or bandwidth-bound). The Transformer's fused JVP kernel is instead bound by on-chip cache bandwidth (Appendix C). *Critical-path.* The step timeline (Figure 1, Appendix D) decomposes the last device's critical path on four devices at $r = 16$ into the chain fill until its construction can start (19.7 ms), its Jacobian construction (44.5 ms), its local backward (19.0 ms), and the end-of-step reductions (3.3 ms). Only the construction depends on the kernels (Section 6).

5 Related Work

The formulation of backpropagation as a composition of Jacobian operators to enable parallel scan was introduced by BPPSA (Wang et al., 2020). However, BPPSA and related advances (Danieli et al., 2023; Pehle & Slotine, 2026) rely on full-rank $d \times d$ Jacobians, making each scan combine intractable at modern hidden-state dimensions, and are further restricted to operations with closed-form Jacobians

	ms (GB)		
	Mamba-3	Transformer	Hybrid
tuned pipeline (GPipe, M=4)	52.0 (2.3)	54.1 (2.7)	53.2 (2.5)
lowest-memory pipeline (1F1B, M=8)	105.6 (1.2)	75.4 (1.4)	97.7 (1.2)
region-parallel, r=16	83.0 (9.9)	167.6 (19.6)	122.5 (15.5)
region-parallel, r=8	59.4 (6.3)	128.4 (11.6)	90.3 (9.1)
region-parallel, r=2	42.5 (4.4)	102.6 (9.2)	67.8 (6.5)
sequential reference (1 device)	36.4 (9.0)	112.0 (9.8)	56.3 (8.8)

Table 3: Wall clock per training step and peak per-device memory, ms (GB), on four H100s for the eight-block stack of the hardware setup. RP rows are checked against the same model's single-device scan engine (relative ℓ_2 error of the full gradient vector 0.002–0.006 at $r \geq 8$). The $r = 2$ rows carry no quality claim (Appendix D).

(e.g. convolutions), precluding application to Transformers and SSMs. Our work bridges this gap by observing that inter-layer communication can be restricted to compact latent interfaces, reducing operator Jacobian size to $r \times r$. Other efforts to break the sequential bottleneck include iterative methods (Fagnou et al., 2025; Lim et al., 2023; Gonzalez et al., 2024; Günther et al., 2020), decoupled approaches (Huang et al., 2025; Jaderberg et al., 2017), and Jacobian-Free Backpropagation (Fung et al., 2022), which trade gradient exactness for throughput via approximations or auxiliary objectives. Pipeline parallel techniques (Gomez et al., 2022; Li & Hoefler, 2021; Huang et al., 2019) improve hardware utilization through micro-batch scheduling while leaving the $O(K)$ algorithmic depth intact, which region parallelism reduces to $O(\log K)$. Training over weak interconnects has been approached on the data-parallel axis by amortizing gradient synchronization (Douillard et al., 2023; 2025), on the model-parallel axis by pipeline schemes tolerant of slow or unreliable links (Ryabinin et al., 2023; Yuan et al., 2022), or by compressing stage-boundary activations (Wang et al., 2022; Ramasinghe et al., 2026). Bounded interfaces make the narrow transport a property of the architecture rather than of the schedule.

6 Discussion & Limitations

Communication and Distributed Inputs. On the shaped link the windowed step at 100 Mbit/s (543 ms) is faster than the tuned pipeline at 1 Gbit/s (1603 ms), at 1 Gbit/s it runs within $2\times$ of its network-free step and is within 5% of it by 3.1 Gbit/s against 77 Gbit/s for the pipeline. And round trips cost 4.1–4.3 ms per millisecond, so a 50 ms intercontinental link adds about 0.2 s. Synchronous training with an exact gradient therefore extends to the links of multi-region cloud training (0.3–2 Gbit/s, (Yuan et al., 2022)) and of decentralized device pools (60–350 Mbit/s, (Ramasinghe et al., 2026; Ryabinin et al., 2023)), which pipeline schemes reach only by quantizing activations to 2–8 bits (Wang et al., 2022; Ramasinghe et al., 2026). With nothing quantized, the windowed step still runs $7.8\times$ faster than a 4-bit-compressed pipeline at 100 Mbit/s. The advantage exists on links below a few Gbit/s, at 10 Gbit/s it is $2.2\times$, and on NVLink the pipeline is faster. These results follow from the architecture rather than from the schedule: every inter-region interface is bounded, so the critical path carries states and $r \times r$ Jacobians whose size is independent of width and region depth, and the only activation-sized transfer is a gradient, which can be accumulated and deferred. A bounded channel carries only what its receiver cannot recompute from its own input, so settings whose inputs are distributed by nature (sensory streams, modalities, or parties) (Lillicrap et al., 2020; Baltrušaitis et al., 2018; Kairouz & McMahan, 2021) are where it has the most to carry, and per-region readouts and input partitions are the extensions of Theorem 2.7's transport most worth pursuing.

Crossover and Headroom. Bounded-interface backpropagation removes the backward's serial depth, $O(\log K)$ in place of $O(K)$ (Proposition 3.1), at $c+1$ forward equivalents of work (with c linear in r), in aggregate compute as well as wall clock (Appendix E). The method reduces sequential depth and communication, not FLOPs. Its wall-clock advantage over sequential backpropagation is therefore a crossover in K, projected at $K \gtrsim c + 1$ from measured components (Section 4.2). At $r = 2$, where the construction is cheapest, the four-device step runs at $1.17\times$ the sequential reference's, approaching the crossover without reaching it (Appendix C). The improvable term is the Jacobian construction c, about half of the last device's critical path on four devices (Figure 1). Halving c by

restructuring the recurrence kernel so that its register and shared-memory footprint no longer caps its occupancy (Appendix C) brings that critical path from 87 to about 65 ms, and the reuse end, $c = g(16) = 1$ (the analytic bound), would bring it to about 46 ms. A faster construction therefore lowers the projected crossover from about $K = 16$ toward $K = 2$. The forward message chain remains sequential in K, the streaming schedule overlaps the Jacobian construction with the chain fill but does not shorten it, so the $O(\log K)$ depth applies to the backward and not to the forward.

Interface Structure and Realization. The formulation exposes the inter-region Jacobians as $r \times r$ objects, so the conditioning of gradient transport across regions, which is not observable in standard backpropagation, can be tracked directly and is an open direction. Theorem 2.7 makes the gradient exact but places no bound on their gains, so a deeper chain could compound them. Restricting the interface transition to a non-expansive family, as manifold-constrained hyper-connections do for residual streams (Xu et al., 2026), would cap them at one by construction. The framework fixes only the r-dimensional channel but not its realization. The two-layer MLP maps of Section 3.1 are one instantiation, a linear projection would cost $\Theta(rD)$ with smaller constants, and structured alternatives (gated linear projections, cross-attention interfaces (Jaegle et al., 2021), Monarch matrices (Dao et al., 2022)) could reduce the overhead. Balancing overhead against Jacobian conditioning remains an open design problem.

Quality Across Scale and Granularity. The gap to dense does not change with the interface rank over the ranges measured (Section 4.1), so the narrowest interface, the cheapest to differentiate, can be used. It does change with the region granularity: it falls as regions coarsen, which puts quality at odds with the span advantage of Proposition 3.1, since that advantage needs $K \gtrsim c + 1$ regions (Section 4.2). On an 18-block model at the current implementation's c no K satisfies both. Deeper models satisfy both, because at fixed K their regions deepen (a 48-block model at $K = 8$ has six-block regions), such that the method's regime is deep models sharded at moderate K. The results cover 47M–321M backend parameters, 2.95B tokens, widths up to 1024, and four devices. Beyond these the gap is untested, the bounded-interface model does not outperform a dense model of the same depth, and conversion of a pretrained dense model by continued training is an open direction.

7 Conclusion

We have introduced Latent Bounded Interfaces (LBI), which reduce the sequential depth of backpropagation from $O(K)$ to $O(\log K)$ by routing inter-region communication through an r-dimensional interface whose adjoint recursion is an exact suffix scan over $r \times r$ Jacobians. The construction costs 14.5 forward equivalents on Mamba-3, and on four devices the step runs at $0.88\times$ a tuned pipeline's speed on NVLink, below the projected crossover. Its inter-region transport is kilobytes per link in place of the pipeline's activation-sized exchanges, so on a shaped link it runs $14\times$ faster at 1 Gbit/s and $30\times$ faster at 100 Mbit/s, where exact-gradient training over multi-region and decentralized links becomes possible. The quality cost is 0.33–0.67 CE on language modeling from 47M to 321M parameters, set by region granularity and unchanged with the interface width over the ranges measured, so the narrowest interface can be used. Bounded interfaces thus give region-parallel training a priced alternative to pipeline parallelism on weak links, with distributed-input settings as its direct continuation.

AI use statement

In this work, we used generative AI tools for implementing methods, providing feedback on research methodology and experiments, refining hypotheses, interpreting results. We have not used generative AI tools for generating synthetic datasets, helping to develop theoretical models or conceptual frameworks, formulate mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, assisting with translation, cleaning and reformatting datasets, supporting qualitative and thematic data analysis. Additionally, we used generative AI tools for suggesting experimental parameters, creating and editing software code, analyzing existing literature, discovering research topics and identifying gaps, sourcing/searching for information, editing the research paper to improve readability, identifying relevant literature. We have reviewed all AI-assisted work. For example, LLM-generated code was verified and tested for correctness throughout the codebase, including but not limited to training code, parallel execution, and GPU kernels. We have taken care to ensure LLM feedback and interpretation is supported by existing literature. We have

also checked all sources found by LLM search by manual literature review for each reference. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Ethics statement

This work is a training-systems method for language models and involves no human subjects, personal data, or new dataset collection. Experiments use the publicly released FineWeb-Edu corpus and LLaMA tokenizer, and the released code contains no model weights or data. We are not aware of applications or societal risks specific to this method beyond those shared by language-model training in general.

Reproducibility statement

Proofs of every result in Sections 2 and 3 are given in Appendix A, and the separator argument behind Definition 2.2 in Appendix B. The architecture is specified in Section 3.1. The training protocol, seeds, learning rates, evaluation procedure, and hardware behind the experiments in the paper are in Appendix E. The timing and network measurement protocols, pipeline baseline, and byte count are in Appendix D. The gradient engines and their parity checks against autograd are in Appendix D. The code for the full implementation is available at `https://anonymous.4open.science/r/latent-bounded-interfaces-33ED/`, with documentation that gives detailed instructions on how to run and reproduce every experiment found in the paper.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2): 423–443, 2018.

Atılım Günes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.*, 18(1):5595–5637, January 2017. ISSN 1532-4435.

Guy E. Blelloch. Prefix sums and their applications. In *Synthesis of Parallel Algorithms*, pp. 35–60. Morgan Kaufmann, 1990.

Guy E Blelloch. Programming parallel algorithms. *Communications of the ACM*, 39(3):85–97, 1996.

Thomas Böhme, Frank Göring, and Jochen Harant. Menger's theorem. *Journal of Graph Theory*, 37 (1):35–36, 2001.

Federico Danieli, Miguel Sarabia, Xavier Suau Cuadros, Pau Rodriguez, and Luca Zappella. DeepPCR: Parallelizing Sequential Operations in Neural Networks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 47598–47625. Curran Associates, Inc., 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/file/948d8ba4e30c8c3a800cf436b31f376e-Paper-Conference.pdf`.

Tri Dao, Beidi Chen, Nimit S Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. Monarch: Expressive structured matrices for efficient and accurate training. In *International Conference on Machine Learning*, pp. 4690–4721. PMLR, 2022.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

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

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

Erwan Fagnou, Paul Caillon, Blaise Delattre, and Alexandre Allauzen. Accelerated training through iterative gradient propagation along the residual path. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=JDm7oIcx4Y.

Samy Wu Fung, Howard Heaton, Qiuwei Li, Daniel McKenzie, Stanley Osher, and Wotao Yin. JFB: Jacobian-Free Backpropagation for Implicit Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6648–6656, 2022.

Aidan N. Gomez, Oscar Key, Kuba Perlin, Stephen Gou, Nick Frosst, Jeff Dean, and Yarin Gal. Interlocking backpropagation: Improving depthwise model-parallelism. *Journal of Machine Learning Research*, 23(171):1–28, 2022. URL http://jmlr.org/papers/v23/20-1121.html.

Xavier Gonzalez, Andrew Warrington, Jimmy T.H. Smith, and Scott W. Linderman. Towards Scalable and Stable Parallelization of Nonlinear RNNs. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 5817–5849. Curran Associates, Inc., 2024. doi: 10.52202/079017-0189. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/0b2b199fdd52089b31d3a0120e400b2a-Paper-Conference.pdf.

Stefanie Günther, Lars Ruthotto, Jacob B. Schroder, Eric C. Cyr, and Nicolas R. Gauger. Layer-Parallel Training of Deep Residual Neural Networks. *SIAM Journal on Mathematics of Data Science*, 2(1):1–23, 2020. doi: 10.1137/19M1247620. URL https://doi.org/10.1137/19M1247620.

Roger W Hockney. The communication challenge for mpp: Intel paragon and meiko cs-2. *Parallel computing*, 20(3):389–398, 1994.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/093f65e080a295f8076b1c5722a46aa2-Paper.pdf.

Zih-Hao Huang, You-Teng Lin, and Hung-Hsuan Chen. Deinforeg: A decoupled learning framework with information regularization for better training throughput. *Neurocomputing*, 651:130813, 2025. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2025.130813. URL https://www.sciencedirect.com/science/article/pii/S0925231225014857.

Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1627–1635. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/jaderberg17a.html.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pp. 4651–4664. PMLR, 2021.

Peter Kairouz and H Brendan McMahan. Advances and open problems in federated learning. *Foundations and trends in machine learning*, 14(1-2):1–210, 2021.

Aakash Lahoti, Kevin Y Li, Berlin Chen, Caitlin Wang, Aviv Bick, J Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved Sequence Modeling using State Space Principles. *arXiv preprint arXiv:2603.15569*, 2026.

Shigang Li and Torsten Hoefler. Chimera: efficiently training large-scale neural networks with bidirectional pipelines. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384421. doi: 10.1145/3458817.3476145. URL https://doi.org/10.1145/3458817.3476145.

Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

Yi Heng Lim, Qi Zhu, Joshua Selfridge, and Muhammad Firmansyah Kasim. Parallelizing non-linear sequential models over the sequence length. *arXiv preprint arXiv:2309.12252*, 2023.

Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R Devanur, Gregory R Ganger, Phillip B Gibbons, and Matei Zaharia. Pipedream: Generalized pipeline parallelism for dnn training. In *Proceedings of the 27th ACM symposium on operating systems principles*, pp. 1–15, 2019.

Pitch Patarasuk and Xin Yuan. Bandwidth optimal all-reduce algorithms for clusters of workstations. *Journal of Parallel and Distributed Computing*, 69(2):117–124, 2009.

Christian Pehle and Jean-Jacques Slotine. Unlocked Backpropagation using Wave Scattering. *arXiv preprint arXiv:2602.10461*, 2026.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 30811–30849. Curran Associates, Inc., 2024. doi: 10.52202/079017-0970. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/370df50ccfdf8bde18f8f9c2d9151bda-Paper-Datasets_and_Benchmarks_Track.pdf.

Sameera Ramasinghe, Thalaiyasingam Ajanthan, Gil Avraham, Yan Zuo, and Alexander Long. Subspace networks: Scaling decentralized training with communication-efficient model parallelism. *Advances in Neural Information Processing Systems*, 38:175147–175166, 2026.

Max Ryabinin, Tim Dettmers, Michael Diskin, and Alexander Borzunov. Swarm parallelism: Training large models can be surprisingly communication-efficient. In *International Conference on Machine Learning*, pp. 29416–29440. PMLR, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Jue Wang, Binhang Yuan, Luka Rimanic, Yongjun He, Tri Dao, Beidi Chen, Christopher Ré, and Ce Zhang. Fine-tuning language models over slow networks using activation quantization with guarantees. *Advances in Neural Information Processing Systems*, 35:19215–19230, 2022.

Shang Wang, Yifan Bai, and Gennady Pekhimenko. Bppsa: Scaling back-propagation by parallel scan algorithm. In I. Dhillon, D. Papailiopoulos, and V. Sze (eds.), *Proceedings of Machine Learning and Systems*, volume 2, pp. 451–469, 2020. URL https://proceedings.mlsys.org/paper_files/paper/2020/file/31f0ece68edf17cf695fdf4e9c5e17d1-Paper.pdf.

Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Binhang Yuan, Yongjun He, Jared Davis, Tianyi Zhang, Tri Dao, Beidi Chen, Percy S Liang, Christopher Re, and Ce Zhang. Decentralized training of foundation models in heterogeneous environments. *Advances in Neural Information Processing Systems*, 35:25464–25477, 2022.

A PROOFS

Proof of Lemma 2.3 (Adjoint Factorization Through Interfaces). Since m_k is sufficient, the loss can be written as $\mathcal{L} = \tilde{\mathcal{L}}(m_k; \theta_{\geq k})$ for some function $\tilde{\mathcal{L}}$ (Definition 2.2). For any variable $w \in \mathcal{R}_{<k}$, the parameters $\theta_{\geq k}$ are independent of w, so $\mathcal{L}$ depends on w only through m_k. By the chain rule, $\partial\mathcal{L}/\partial w = (\partial m_k/\partial w)^\top \, \partial\mathcal{L}/\partial m_k$. Since this holds for every $w \in \mathcal{R}_{<k}$, Eq. 1 follows. □

Proof of Corollary 2.6 (Sequential Interface Representation). Fix $k \in \{0, \ldots, K-1\}$. By sufficiency at boundary k (Definition 2.2), every node in $\mathcal{R}_{\geq k}$ (in particular every node internal to $\mathcal{R}_k$) depends on $\mathcal{R}_{<k}$ only through m_k. The outgoing interface state m_{k+1} is computed from nodes in $\mathcal{R}_k$ together with the incoming interface m_k. Since all of these depend on $\mathcal{R}_{<k}$ only through m_k, the state m_{k+1} is determined by m_k and the local parameters θ_k alone, yielding the transition map 2. Applying this argument inductively for $k = 0, \ldots, K-1$ produces the interface chain $m_0 \to m_1 \to \cdots \to m_K$. By Lemma 2.3, all reverse-mode adjoints across region boundaries also factor through this chain. □

Proof of Theorem 2.7 (Bounded-Interface Backpropagation). Fix $k \in \{0, \ldots, K-1\}$. By Corollary 2.6, all influence of earlier regions on later computation factors through the interface chain, and in particular through m_{k+1}. Applying Lemma 2.3 at boundary $k{+}1$ gives $\bar{m}_k = (\partial m_{k+1}/\partial m_k)^\top \bar{m}_{k+1}$. A second application of the chain rule to the local parameters θ_k, whose influence on $\mathcal{L}$ factors solely through m_{k+1}, gives $\nabla_{\theta_k}\mathcal{L} = (\partial m_{k+1}/\partial \theta_k)^\top \bar{m}_{k+1}$. Finally, unrolling the recursion of $\bar{m}_k$ yields $\bar{m}_k = J_k^\top \bar{m}_{k+1} = J_k^\top J_{k+1}^\top \bar{m}_{k+2} = \cdots = \left(\prod_{j=k}^{K-1} J_j^\top\right) \bar{m}_K$, proving 3. For $k = K$, the product is empty and therefore equals identity. □

Proof of Proposition 2.8 (Scan Formulation of Backpropagation). By Theorem 2.7, the inter-region adjoints satisfy $\bar{m}_k = \left(\prod_{j=k}^{K-1} J_j^\top\right) \bar{m}_K = P_k \bar{m}_K$. Hence, computing all inter-region adjoints reduces to computing the suffix products $\{P_k\}_{k=0}^{K}$ of the sequence $\{J_k^\top\}_{k=0}^{K-1}$. Since matrix multiplication is associative, these suffix products can be computed by any standard parallel prefix/suffix scan algorithm (Blelloch, 1990). □

Proof of Corollary 2.9 (Region-Local Backward Independence). By Theorem 2.7, the contribution of region k to the backward pass factors as $\nabla_{\theta_k}\mathcal{L} = (\partial m_{k+1}/\partial \theta_k)^\top \bar{m}_{k+1}$. The Jacobian $\partial m_{k+1}/\partial \theta_k$ depends only on the forward computation internal to region k, and therefore only on region-k forward activations and local parameters θ_k. It does not depend on activations, parameters, or adjoints from any other region, except through the incoming interface adjoint $\bar{m}_{k+1}$. Consequently, once the interface adjoints have been computed, each region-local backward computation depends only on local data and may be executed independently across regions. □

Proof of Proposition 3.1 (Work–Span Decomposition). By Proposition 2.8, inter-region adjoint transport reduces to a suffix scan over interface Jacobians J_k, contributing total work $\Theta(Kr^3)$ and span $\Theta(r^3 \log K)$ for uniform r. Constructing each J_k requires propagating r perturbation directions through the linearized region R_k, at a cost we denote W_k^J (quantified below). Since these constructions depend only on region-local forward caches and the incoming m_k, all K constructions are independent, contributing total work $\sum_k W_k^J$ and span $\max_k W_k^J$. By Corollary 2.9, once the

interface adjoints $\{\bar{m}_k\}$ are available, all region-local backward computations are independent, contributing total work $\sum_k W_k^{\text{local}}$ and span $\max_k W_k^{\text{local}}$. The three components execute sequentially: Jacobian construction requires the forward caches, the scan requires the constructed Jacobians, and region-local backward requires the scan output. □

B VERTEX SEPARATORS

Definition B.1 (Interface Separator)**.** A set of nodes $S_k \subset \mathcal{V}$ is an *interface separator* at boundary k if every directed path from $\mathcal{R}_{<k}$ to $\mathcal{R}_{\geq k}$ contains at least one node in S_k.

Lemma B.2 (A Separator Implies Sufficiency)**.** *If S_k is an interface separator at boundary k, then the vector m_k formed by collecting the values of nodes in S_k is sufficient in the sense of Definition 2.2.*

Proof. We proceed by induction on the topological order of $\mathcal{G}$, restricted to $\mathcal{R}_{\geq k}$. Let $v \in \mathcal{R}_{\geq k}$ and suppose the claim made by the lemma holds for every node in $\mathcal{R}_{\geq k}$ that precedes v in topological order. If $v \in S_k$, then the value of v is a component of m_k and the claim immediately holds. If $v \notin S_k$, then consider the parents of v in $\mathcal{G}$. Notice that each parent u of v satisfies exactly one of the following conditions:

1. $u \in \mathcal{R}_{\geq k}$. Then, u precedes v in topological order, so by the inductive hypothesis, u is a function of $(m_k, \theta_{\geq k})$.
2. $u \in \mathcal{R}_{<k}$. Then the edge $u \to v$ is a directed path from $\mathcal{R}_{<k}$ to $\mathcal{R}_{\geq k}$, which must contain a node in S_k by Definition B.1. Since this path consists of only the two nodes u and v, and $v \notin S_k$, we conclude that $u \in S_k$. Therefore the value of u is a component of m_k.

In either case, the value of every parent u of v is determined by $(m_k, \theta_{\geq k})$. Since v computes a deterministic function of its parents, v is also a function of $(m_k, \theta_{\geq k})$, which completes the induction. □

Remark B.3 (Minimum Separator Size)*.* By Menger's vertex theorem (Böhme et al., 2001), the minimum vertex separator between $\mathcal{R}_{<k}$ and $\mathcal{R}_{\geq k}$ in a standard feedforward computational graph has size d_k, which is the full hidden state since each scalar component admits an independent directed path across the boundary. The bounded-interface property (Definition 2.4) therefore requires architectural modification. The encoder/decoder projections of Section 3.1 create a new vertex separator of size $r_k \ll d_k$ in the modified graph, which cannot be achieved by selecting a subset of existing nodes.

Graph-Theoretic vs. Effective Minimality. Remark B.3 establishes that exact sufficiency formally requires the full hidden state d_k. Interfaces of dimension $r \ll d_k$ train within 0.33–0.67 CE of dense (Section 4). The shared canvas accounts for the small information deficit, since it carries base token information to every region so the interface encodes only inter-region refinements. A theoretical characterization of the effective interface rank remains open.

C COST MODEL AND CONSTRUCTION CONSTANT

We place the measured construction constant against the reuse model of Section 3.2 and project the crossover.

Scan Cost. The suffix scan over K interface Jacobians of size $r \times r$ has total work $\Theta(Kr^3)$, $9 \times 16^3 \approx 3.7 \times 10^4$ FLOPs at the shape of Table 1, against a Jacobian construction of $c \approx 14.5$ forward passes (Section 4.2), and does not appear as a separate term in the measured critical path.

Measured Placement. The bracket of Section 3.2, $g(r) \leq c \leq r$ with $g(16) = 1$ for the SSM region (its forward intensity sits far below the device ridge), puts the fused implementation near its no-reuse end ($c = 14.5$ on the 120M configuration, Section 4.2). Hardware performance counters (Table 4) place the recurrence kernel of the chunked SSM scan far from either roofline: SM issue, DRAM, and L1 throughput all sit below half their ceilings. Its register and shared-memory footprint caps occupancy at 12.5%, so the scheduler has no other warp to issue while one waits on the serial

kernel	occupancy	SM	DRAM	L1/shared	L2 hit
Mamba-3 recurrence	12.5%	32%	15%	46%	18%
Transformer fused attention JVP	—	—	12%	70%	79%

Table 4: Hardware counters of the construction kernels (H100, ncu, throughputs as a fraction of peak): the recurrence kernel of the chunked SSM scan on the 120M Mamba-3 configuration of Table 1, and the Transformer's fused attention JVP. Dashes are counters not read for that kernel.

chunk-to-chunk dependency, and the kernel's time is set by that chain's latency rather than by the arithmetic it issues. The idle arithmetic units are the capacity the reuse model assumed would absorb the tangent directions. The chunked-scan and peripheral kernels, which carry the r directions as a batch, are not memory-bound at this shape either, so their time grows linearly with the direction count instead of being absorbed into the forward's memory traffic, which is the constant per-direction cost the $c(r)$ sweep measures (Section 4.2). The Transformer kernel sits at the other end of the bound: its regions are compute bound ($I_{\mathrm{fwd}} \geq I_{\mathrm{ridge}}$), so $g(r) \propto r$, and the measured construction is direction-linear at the four-device shape ($c = 0.98/1.68/3.31/6.75$ at $r = 2/4/8/16$, $\approx 0.42r$). Its fused attention JVP is bound by on-chip bandwidth, with tangent operands cache-resident, rather than by the compute the reuse analysis projects.

Kernel Breakdown. Kernel-level times (nsys) over two full Jacobian constructions on the 120M configuration ($K = 9$ regions), every kernel the construction launches: the two kernels of the chunked SSM scan take 61% of construction GPU time, the intra-chunk scan 31% and the serial recurrence kernel of Table 4 30%, and the peripheral kernels (encoder/decoder projections, normalizations, rotary embedding, pointwise work) the remaining 39%. Summed kernel time is ≈ 912 ms per construction against the 905 ms wall measurement, so the decomposition accounts for the construction to within scheduling gaps. The construction wall clock is likewise a property of the region kernel and the token count rather than of sequence shape. Halving the batch at doubled context reproduces the batch-32 construction to 1.1% (915.0 ms against 904.7 ms).

Crossover Projection. With c measured, the span comparison of Proposition 3.1 projects the wall-clock crossover against sequential backpropagation at $K \approx c + 1$: $K \approx 16$ at $r = 16$ for Mamba-3 and $K \approx 9$ at $r = 8$ (the construction is linear in r). At $r = 2$, where the construction is the cheapest ($c(2) = 2.89$, construction is 180 ms against the 62.3 ms forward at the single-device shape), the projected crossover is $K \approx 3.9$, and the four-device step (42.5 ms) is within $1.17\times$ of the one-device step (36.4 ms) without passing it.

D Systems and Measurements

Algorithm 1 presents bounded-interface backpropagation as three strictly sequential phases: Jacobian construction, scan composition, and region-local backward. This ordering is correct but still conservative, and does not exploit the fact that Jacobian construction for region k depends only on region k's forward cache C_k and the incoming interface state m_k, both of which are available as soon as region k's forward pass completes. The streaming schedule overlaps Jacobian construction with the sequential forward pass: each region's construction is launched asynchronously as soon as its forward completes, the scan waits for all of them, and the rest of Algorithm 1 is unchanged.

The overlap window depends on the relative cost of the forward pass and the Jacobian construction, $\Theta(F_k)$ against $\Theta(r_k F_k)$ per region (Eq. 6): construction for region k proceeds concurrently with the forward passes of regions $k{+}1, \ldots, K{-}1$ and completes inside them only when $K \gg r_k$. Figure 1 shows the measured overlap at $K = 4$. A second overlap opportunity exists between the scan and the region-local backward: once the suffix scan produces $\bar{m}_k$, that region's local backward can begin immediately. The region-parallel implementation exploits both overlaps, where device k owns region k, Jacobian construction for earlier regions proceeds while the forward message chain is still filling through later ones, and the seeded per-region local backward begins as each adjoint arrives. Table 3 reports the measured wall clock of this schedule on four devices, and Figure 1 its kernel-level timeline.

Algorithm 1 Bounded-Interface Backpropagation

Require: Region transition maps $\{R_k\}_{k=0}^{K-1}$, forward caches $\{C_k\}_{k=0}^{K-1}$, local parameters $\{\theta_k\}_{k=0}^{K-1}$, terminal interface adjoint $\bar{m}_K$

Ensure: Interface adjoints $\{\bar{m}_k\}_{k=0}^{K}$ and parameter gradients $\{\nabla_{\theta_k}\mathcal{L}\}_{k=0}^{K-1}$

▷ Phase 1: Jacobian construction

for all regions $k = 0, \ldots, K-1$ **in parallel do**

  Construct interface Jacobians $J_k = \frac{\partial m_{k+1}}{\partial m_k} \in \mathbb{R}^{r\times r}$ from C_k via AD

end for

▷ Phase 2: Interface adjoint scan

Compute suffix products $\{P_k\}_{k=0}^{K}$ via parallel scan over $\{J_k^\top\}_{k=0}^{K-1}$

for all $k = 0, \ldots, K$ **in parallel do**

  $\bar{m}_k \leftarrow P_k \bar{m}_K$

end for

▷ Phase 3: Region-local backward

for all regions $k = 0, \ldots, K-1$ **in parallel do**

  Compute $\nabla_{\theta_k}\mathcal{L} = \left(\frac{\partial m_{k+1}}{\partial \theta_k}\right)^\top \bar{m}_{k+1}$ using C_k

  Compute any additional local adjoints for region k

end for

return $\{\bar{m}_k\}_{k=0}^{K}$ and $\{\nabla_{\theta_k}\mathcal{L}\}_{k=0}^{K-1}$

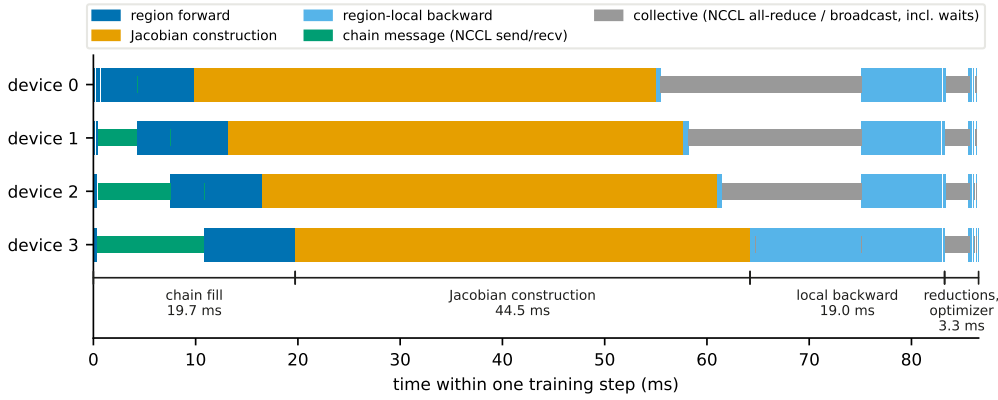

Figure 1: Kernel-level timeline of one region-parallel training step ($K = 4$ devices, Mamba-3, $r = 16$, from the four-device measurement of Table 3), anchored at the start of device 0's forward. Jacobian construction begins on each device as its own region's forward completes, while the message chain is still filling through later devices (onsets at 9.9 / 13.2 / 16.5 / 19.7 ms on devices 0–3). The chain messages themselves cost no exposed time, and devices 0–2 then wait in the scan's collectives until the last device's construction ends. The bracket gives the last device's critical path.

Implementation and AD. Our implementation provides three interchangeable engines: (i) the *forward-mode construction*, fused kernels that propagate all r tangent directions through each region in one pass (the chunked-scan and direction-batched recurrence kernels for the SSM, a flash-attention JVP for the Transformer); (ii) a *native reverse construction* that extracts $J_k^\top$ columns by VJPs against the basis from frozen per-region caches; (iii) the *autograd graph construction*, the reference. The interface maps of Section 3.1 are differentiated by the same direction-batched forward mode and verified against the reference by adjointness and parity checks. Every engine computes the interface-to-interface Jacobian $J_k = \partial m_{k+1}/\partial m_k$ only, and parameter gradients are left to the region-local backward, a native seeded VJP (Corollary 2.9). The scan is seeded at $\bar{m}_{K-1}$, the loss pulled back through the terminal region, and composes $J_0^\top, \ldots, J_{K-2}^\top$. The terminal region's parameters take the loss's cotangent directly. The training runs of Table 1 use the autograd engine. By these checks the engine choice moves gradients only at the bf16 noise floor, and trained end-to-end the engines match

at the run level: 3.504 CE (scan) against 3.500 CE (autograd) on the 15k-step truncated schedule, and within 0.008 CE of the autograd seeds at the full budget on all three backbones (post-hoc, with the Hybrid run token-matched at half batch).

Measurement Protocol. Single-device engine and construction rows are timed over 10 iterations after 3 warmup iterations, with device synchronization at the boundaries. Four-device rows are timed over 5 iterations after 3, each iteration between a device synchronization and a barrier, in a single run on a $4\times$H100 NVL node. Window rows are timed around the whole block of N steps that contains one window collective, with the reported per-step time divided by N. Every timed step is a forward and backward only and no optimizer step is included. The region-parallel gradient is checked against the same model's scan engine on one device by the relative ℓ_2 error of the full gradient vector, with the worst per-parameter max-abs ratio reported alongside.

Pipeline Baseline. The four-device pipeline uses torch's ScheduleGPipe and Schedule1F1B over four stages of two blocks. The schedule requires at least as many microbatches as stages, so $M = 4$ is its smallest configuration. Its step depends on the microbatch count through the backbone's kernel efficiency at small batch. Table 3 reports the fastest and the lowest-memory configurations: GPipe at $M = 4$ takes 52.0 / 54.1 / 53.2 ms per step on Mamba-3 / Transformer / Hybrid, against 99.5–105.6 ms at $M = 8$ under either schedule and 60.0 / 54.2 / 57.4 ms under 1F1B at $M = 4$. 1F1B at $M = 8$ holds 1.2 / 1.4 / 1.2 GB against GPipe's 2.3 / 2.6 / 2.4 GB. The Hybrid's pipeline step equals the pure-SSM stack's because its slowest stage is its SSM-only stage. With two stages per device in interleaved placement (eight one-block stages, stage s on device $s \bmod 4$, torch's looped-BFS and interleaved-1F1B schedules), the pipeline counts 469.8 MB per step, $2 \times 7 \times 33.55$ MB, against 201.3 MB with one stage per device. At width 2048 the tuned pipeline takes 1185 ms at 1 Gbit/s against 151 ms for the 32-step window ($7.8\times$). On the Hybrid at width 1024 the tuned pipeline and the 32-step window take 644 / 103 ms at 1 Gbit/s and 5478 / 216 ms at 100 Mbit/s (emulated, medians of three).

configuration	step (ms)	peak memory (GB)
region-parallel, one pass (Table 3)	83.0	9.9
construction chunked over 8 directions	86.2	6.3
over 4 directions	93.2	4.5
over 2 directions	109.2	4.4
4 microbatches, chunks of 4	134.9	2.5
8 microbatches, chunks of 4	223.2	2.2
tuned pipeline (GPipe, M=4)	52.0	2.3
lowest-memory pipeline (1F1B, M=8)	105.6	1.2
sequential reference (1 device)	36.4	9.0

Table 5: Peak per-device memory against step time on Mamba-3 at $r = 16$ at the shape of Table 3: the one-pass configuration (all r tangent directions at once), the Jacobian construction chunked over 8 / 4 / 2 directions, and four and eight microbatches with chunks of four, against the two pipeline configurations and the sequential reference. Chunking and microbatching leave the gradients unchanged in exact arithmetic. the state a device holds across the step is 1.7 GB, the rest of each peak is transient.

The $r = 2$ rows. Table 3 reports $r = 2$ to show the wall-clock mechanism (construction linear in r) without a quality claim. Quality is flat from $r = 8$ (Table 9), and the LayerNorm close of Eq. 5 has a rank-zero differential at $r = 2$ (LayerNorm is invariant to shift and scale, so its differential has rank at most $r - 2$), thus the $r = 2$ rows time the implementation on a degenerate transport.

Network Experiment. The byte count behind Table 2 is taken at the torch.distributed call boundary per rank per step and classified as critical-path (Section 4.2) or deferrable. All-reduce is counted as its ring wire bytes, $2(K-1)/K$ times the payload (Patarasuk & Yuan, 2009). The latency–bandwidth bound of Section 4.2 needs only the byte count and the network-free step T_0 and is independent of any schedule or overlap. It holds to within 5% below 1 Gbit/s (Section 4.2) and not at 10 Gbit/s, where it is decided by measurement. In the window rows the shared interface scalar's partial gradients are accumulated over the window and reduced at its boundary with the canvas gradient. A step's

serial chain is then the $K-1$ messages and the loss-adjoint broadcast, with the Jacobian all-reduce issued as each device's construction ends and overlaps the chain. Reducing the scalar every step adds one collective and raises the round-trip slope from 4.1–4.3 to 6.0–6.2 ms per millisecond of RTT (Table 7). The window rows update that scalar once per window; the trained window runs of Table 10 update it every step.. On the emulated link (Table 6) the bound lies within 30 ms of every entry, and the pipeline's step is unchanged when the emulator instead charges each transfer at its wait so that it can overlap compute (643 ms at 1 Gbit/s either way), because torch's GPipe and 1F1B schedules wait on every send batch immediately after issuing it. The emulator is used for bandwidth only: it charges one RTT per operation, which overstates the pipeline's latency exposure where its bytes saturate the link and understates the region-parallel step's several hops per collective, so round-trip sensitivity is measured on the shaped link (Table 7).

The shaped link's rates and round trips follow the slow-network literature: traffic-control bandwidth from 100 Mbit/s to 10 Gbit/s and netem latency (Wang et al., 2022; Ryabinin et al., 2023), and measured cross-region links of 0.3–2 Gbit/s at 5–250 ms and 60 –350 Mbit/s (Yuan et al., 2022; Ramasinghe et al., 2026). The bandwidth to 95% statistic (the link rate at which the step is within 5% of its network-free step) follows Douillard et al. (2025) utilization, whose reported values are from a simulator. Ours are from the emulated sweep at RTT 0.1 ms, which isolates the bandwidth term. The compression proxy scales the pipeline;s point-to-point bytes by 0.25 (4-bit) with quantization work and error left out, which is optimistic for the pipeline (Wang et al. (2022)'s operating point is 2–4 forward and 4–8 backward bits).

scheme / Mbit/s	50 M	100 M	300 M	1 G	3 G	10 G	25 G	100 G
RTT 5 ms								
pipeline GPipe $M{=}4$	10,846	5,477	1,897	643	283	154	122	108
pipeline, bytes $\times 0.25$	2,790	1,447	551	237	145	114	106	103
region-parallel, every step	8,122	4,092	1,406	470	199	105	79	63
region-parallel, window 8	1,065	562	226	109	75	63	61	61
region-parallel, window 32	312	186	102	73	64	61	61	61
RTT 0.1 ms								
pipeline GPipe $M{=}4$	10,805	5,439	1,860	606	243	118	81	63
region-parallel, every step	8,103	4,070	1,380	439	183	72	59	59
region-parallel, window 8	1,059	556	220	103	69	58	57	57
region-parallel, window 32	307	182	98	68	60	57	57	57

Table 6: The emulated-link grid behind Table 2 (ms per step, Mamba-3, $r = 8$, 4 H100s). RTT-5 ms rows: medians of three runs (spread ≤ 3 ms, ≤ 13 ms for the every-step row). RTT-0.1 ms rows: single runs, with about 0.5 ms of host-sleep granularity per transfer.

	1 Gbit/s, RTT 5 / 20 / 50 ms			10 Gbit/s, RTT 5 / 20 / 50 ms		
tuned pipeline	1,603	1,573	1,640	158	327	1,076
region-parallel, window 32	113	174	297	72	134	264

Table 7: Round-trip sensitivity on the shaped link (ms per step, medians of two or three runs, window 32 with block timing, the shared interface scalars reduced in the window collective).

E Training Protocol and Quality

Protocol. Every run of Tables 1 and 11 trains on 2.95B tokens. The granularity curve of Table 12 trains at batch 16 for 90k steps, 1.47B tokens, against a dense reference at the same batch. At the 120–157M scale of Table 1 this is 19–25 tokens per parameter (Hoffmann et al., 2022), and the other scales of Table 11 keep the token budget rather than the ratio. The Transformer models of Table 1 train at batch 16 for 180k steps, token-matched, because the batch-32 bounded-interface configuration exceeds device memory. Intra-region parameters are held fixed rather than matched on totals, so that weaker per-region computation is not conflated with the interface bottleneck. Cross-architectural comparisons of the absolute gap are therefore observational, since layer counts and widths differ across families. Learning rates are tuned per point over a small range on the first 15k steps of the 90k-step cosine schedule, so the tuning run follows the longer run's learning-rate trajectory. The remaining optimizer settings transfer from the dense baselines unchanged.

Gradient Parity. The scan-based backward's full parameter-gradient vector is compared against autograd on the same bounded-interface model, on reduced instances (width 512, $r = 16$, two-block regions, context 512, batch 4) with K from 2 to 14 and ten seeds of initialization and batch per entry, under the forward-mode construction (the graph construction gives the same medians within 25%). Table 8 reports the worst case over K and seeds and the median relative error per K. In bfloat16 the error grows sublinearly with depth on Mamba-3 and the Transformer ($1.7\times$ over a $7\times$ increase in K) and about linearly on the Hybrid ($6.5\times$). In float32 (the Transformer under the graph construction, the fused kernels are bfloat16-only) the discrepancy is below 3×10^{-7} at every K. The bfloat16 growth is therefore reduced-precision materialization along the chain, and not intrinsic to the construction or the scan.

backend	dtype	max-abs	rel. ℓ_2	cosine
Mamba-3	bfloat16	2.7×10^{-3}	1.2×10^{-2}	≥ 0.99994
Transformer	bfloat16	1.2×10^{-2}	1.1×10^{-2}	≥ 0.99995
Transformer	float32	2.4×10^{-7}	2.7×10^{-7}	1.00000
Hybrid	bfloat16	5.6×10^{-1}	8.1×10^{-2}	≥ 0.9967
		median rel. ℓ_2 at $K = 2$ / 4 / 7 / 10 / 14		
Mamba-3	bfloat16	.0058 / .0065 / .0075 / .0083 / .0099		
Transformer	bfloat16	.0040 / .0046 / .0054 / .0067 / .0068		
Transformer	float32	$< 3 \times 10^{-7}$ at every K		
		median rel. ℓ_2 at $K = 2$ / 4 / 8 / 10 / 14		
Hybrid	bfloat16	.0062 / .0093 / .0178 / .0249 / .0402		

Table 8: Gradient parity of the scan-based backward against autograd. Top: worst case over $K \in \{2, \ldots, 14\}$ and ten seeds. Max-abs and relative ℓ_2 error of the full parameter-gradient vector, and its cosine to the reference. Bottom: median relative ℓ_2 error over seeds at each K.

	truncated			full budget	
rank	Mamba-3	Transformer	Hybrid	Mamba-3	Hybrid
$r = 8$	3.516	3.403 ± 0.058	3.279 ± 0.007	3.212 ± 0.002	3.079
$r = 16$	3.500	3.396 ± 0.058	3.267 ± 0.012	3.212 ± 0.002	3.077 ± 0.001
$r = 32$	3.505	3.386 ± 0.056	3.275 ± 0.005	3.211 ± 0.001	—
$r = 64$	3.502	3.382 ± 0.066	3.266 ± 0.004	—	—

Table 9: Interface rank. Truncated: best validation cross-entropy over the first 15k steps of each backend's schedule, one seed, learning rate 6×10^{-4} (Mamba-3 and the Transformer at batch 16, the Hybrid at batch 32). Full budget: same protocol as Table 1, post-hoc, mean $\pm$ standard deviation over three seeds (the Hybrid at $r = 8$, one seed).

configuration	truncated	full budget
no window (Table 1)	3.500	3.212 ± 0.002
$N = 8$, $\times 1$	3.527	—
$N = 8$, $\times 8$	3.509	—
$N = 32$, $\times 1$	3.782	—
$N = 32$, $\times 8$	3.510	3.214 ± 0.001
$N = 32$, $\times 32$	3.525	—

Table 10: The N-step window on Mamba-3 (18L/1024d) at $r = 16$, with the canvas group's learning rate scaled by the multiplier shown. Validation CE on the truncated schedule (as in Table 9) and at the full budget (post-hoc, the window row mean $\pm$ standard deviation over two seeds). The first row is a reference without a window.

Rank. Under the protocol of Table 1 the gap to dense is 0.54 CE at $r = 8$, 16, and 32 (Table 9), and the Hybrid at $r = 8$ (seed 7) trains to 3.079 against 3.077 ± 0.001 at $r = 16$: the rank the

hardware section measures at trains at the same gap. On the truncated schedule the sweep is flat from $r = 8$ to 64 on all three backbones, the rank moves the mean by 0.016 / 0.021 / 0.013 on Mamba-3 / Transformer / Hybrid, against seed-to-seed spreads of about 0.03, 0.08, and 0.01. On Mamba-3 it is flat with the learning rate tuned per rank and equally flat at a fixed rate (3.500–3.516 at 6×10^{-4}; 3.511–3.526 at 1.2×10^{-3}).

Window Quality. Under an N-step window the trainer applies the canvas gradient once per N steps as the window mean, leaving the canvas parameters untouched in between. The region-parallel step reduces the window sum (under AdamW the mean and the sum differ only through ϵ). The canvas group then takes N-fold fewer optimizer steps at effective batch $\times N$, so its learning rate is scaled as for a larger batch by a multiplier on that parameter group. Unadjusted, the window lags the reference on the truncated schedule ($+0.027$ at $N = 8$, $+0.28$ at $N = 32$). With the rate scaled $\times 8$ both windows sit within 0.01 of it (a third of seed noise), and at the full budget the 32-step window trains to 3.214 and 3.213 over two seeds against Table 1's 3.212 ± 0.002 (seeds 3.210–3.214, Table 10).

backend	scale	K	dense	bounded-interface	gap	seeds
Transformer	12L/512d (47M)	6	2.847	3.209	0.36	1
Mamba-3	14L/768d (54M)	7	2.788 ± 0.004	3.294 ± 0.001	0.51	3
Hybrid	12L/768d (61M)	6	2.749 ± 0.003	3.152 ± 0.001	0.40	3
Mamba-3	18L/1024d (120M)	9	2.669 ± 0.001	3.212 ± 0.002	0.54	3
Mamba-3	48L/1024d (321M)	24	2.563 ± 0.032	3.209 ± 0.002	0.65	2

Table 11: Quality at the full budget across scales, same protocol as Table 1 (2.95B tokens, $r = 16$, two-block regions), post-hoc validation cross-entropy, mean $\pm$ standard deviation over the seeds shown. The gap is bounded-interface minus dense. The two 321M seeds give gaps of 0.63 and 0.67.

blocks per region	K	bounded-interface	gap	seeds
2	9	3.271 ± 0.001	0.53	3
3	6	3.135 ± 0.001	0.39	3
6	3	2.969 ± 0.001	0.23	3
9	2	2.906 ± 0.004	0.16	3

Table 12: The region-granularity curve on the 18-block Mamba-3 stack (batch 16, 90k steps, learning rate 3×10^{-4}, $r = 16$), post-hoc validation CE against 2.742 ± 0.003 for the same stack trained dense at the same batch (three seeds). Mean $\pm$ standard deviation over the seeds shown.

Scale and Region Granularity. Table 11 extends the Mamba-3 column of Table 1 to 54M and 321M backend parameters ($K = 7, 24$) and adds the 47M Transformer and the 61M Hybrid, all at the same token budget. On Mamba-3 the gap rises with scale but stays within 0.17 CE from 54M to 321M, and across all backends and scales it spans 0.33–0.67 CE. Table 12 gives the curve behind Section 4.1 on the 18-block stack: the gap falls from 0.53 CE at two-block regions to 0.16 CE at nine-block regions, all four points at 3×10^{-4} learning rate (the nine-block point's tuned rate, 6×10^{-4}, is unstable over the full run). In both tables the gap moves with the region count, by 0.14 CE from $K = 7$ to 24 across scales at fixed two-block regions, and by 0.37 CE from $K = 2$ to 9 on one stack.

Matched Total Parameters. A dense Mamba-3 matched on total parameters instead of depth (21 blocks, 140.6M against the bounded-interface 140.8M, learning rate tuned on the truncated schedule) trains to 2.643 CE at 120M: matching totals improves the dense model and widens the gap to 0.57 CE, and since the bounded-interface model of Table 1 already carries more parameters than its depth-matched comparator, the gap is not a capacity effect. At matched compute the dense model wins: the forward-mode step costs $5.9\times$ the dense step on one device and the four-device step spends $9.1\times$ its device-milliseconds (Section 4.2, Table 3).