JACOBI RECIRCULATION: PARALLELIZING RECIRCULATION WITH FIXED-POINT ITERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A growing line of work lets a transformer reuse its own deeper representations in later computation, increasing its effective depth without adding parameters. One recent approach is recirculation, which feeds the state of each token at a deeper layer back into a shallower layer, thereby rewriting the key–value cache for later tokens to attend to. However, this token-wise recursion makes processing a known sequence expensive in prompt prefill, evaluation, hyperparameter selection, and the training of its auxiliary mixer. In this work, we propose Jacobi recirculation, which replaces the recursion with a few rounds of parallel updates over all token positions. We prove that the iteration is exact after finitely many rounds, converges geometrically under a local contraction condition, and yields convergent training gradients under additional regularity. Experiments across model families and sizes show that two to three rounds suffice to reproduce sequential recirculation, even though the pretrained models were never trained for this iteration. When used for prompt prefill before generation, it also preserves downstream accuracy. In our experiments, it speeds up single-sequence prefill by more than an order of magnitude and reduces mixer training from hours to minutes. Together, these gains make recirculation substantially cheaper to use and to study. Code is available.

1 INTRODUCTION AND RELATED WORK

Scaling test-time computation has become a major route to improving pretrained language models, most visibly through longer chains of thought (Wei et al., 2022; Snell et al., 2025; Guo et al., 2025). A growing family of methods pursues the same goal within the network. Looped and recurrent-depth transformers reuse a block of layers to increase effective depth without adding parameters (Dehghani et al., 2019; Giannou et al., 2023; Saunshi et al., 2025; Geiping et al., 2025), and latent-reasoning models feed high-layer representations back as inputs to later positions (Hao et al., 2025; Wu et al., 2025; Zeng et al., 2026; Huang et al., 2026; Wang et al., 2026; Cai et al., 2026). Recirculation (Mozer et al., 2026) is a recent member of this family. It mixes the deeper-layer representation of each token back into a lower layer and rewrites that token's key–value cache, so that later tokens can attend to its deeper representation. Most feedback architectures pass the high-layer state of one token to later tokens through new connections, which must be trained (Fan et al., 2020; Hao et al., 2025). Recirculation instead feeds the state of each token back into the same token, so it can be applied to a frozen pretrained model, optionally with a small trained mixer that sets the mixing coefficients. It has been reported to improve language modeling, instruction following, and reasoning.

However, recirculation is expensive to run. Each token must read the cache rewritten by earlier tokens, so T tokens take T sequential steps. This makes recirculation slow in settings where a standard transformer processes all positions in parallel, such as prompt prefill, evaluation, and mixer training. Recirculation is also sensitive to its layer pair and mixing coefficient, so selecting them requires many such evaluations. This cost limits both the use and the study of recirculation.

In this work, we address this problem by replacing the sequential computation with Jacobi-style iterations that update all token positions in parallel (Figure 1A–B). Parallel fixed-point iterations have long been used to parallelize sequential computation, from feedforward networks and nonlinear recurrences (Song et al., 2021; Lim et al., 2024; Gonzalez et al., 2024) to the decoding of language models (Santilli et al., 2023; Fu et al., 2024; Kou et al., 2024). Latent-reasoning models use a similar idea to approximate their sequential feedback (Wu et al., 2025; Zeng et al., 2026; Huang

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.

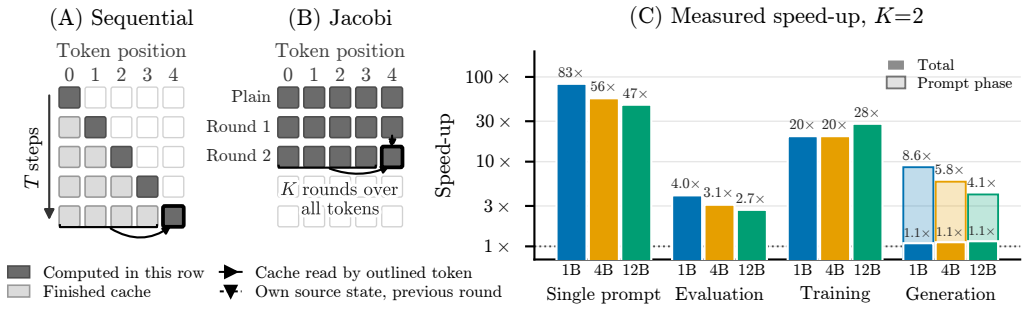


Figure 1: (A) Sequential recirculation processes one token at a time, each reading the rewritten cache of all earlier tokens. (B) Jacobi recirculation starts from a plain forward pass and updates all tokens in parallel for K rounds, each reading the cache that earlier tokens write in the same round. (C) Wall-clock speed-up with $K = 2$ on AMD MI210 GPUs for single-sequence prefill and the scenarios of Section 4.1. For generation, light bars show the prompt phase and dark bars the total time, which is dominated by token-by-token decoding. Sequential recirculation reloads the weights at every token step and is *memory-bound*, whereas Jacobi recirculation amortizes each weight load over the sequence and is *compute-bound*. The gain is therefore largest for a single prompt and for training, where each device processes only a few sequences, and smaller for evaluation, which processes many sequences at once (Section 4.2).

et al., 2026; Wang et al., 2026; Cai et al., 2026). They train through a few parallel rounds instead of the exact recursion. However, it is unknown at scale how well a few rounds approximate the exact recursion during training. This question becomes critical when the idea is used to accelerate recirculation. Since recirculation leaves the pretrained model unchanged, the effect of the method depends entirely on how well the rounds approximate the sequential computation. We answer this question for recirculation theoretically and empirically, and find that two to three rounds suffice for the pretrained models we study.

Contributions. Our main contributions are as follows. (1) We formulate recirculation as a fixed-point problem whose causal structure makes Jacobi recirculation exact after finitely many rounds. We then derive a sufficient condition for geometric convergence from layerwise sensitivity bounds on the frozen transformer and the mixer, and bound the error of training gradients backpropagated through a finite number of rounds. (2) We show empirically that two to three rounds of Jacobi recirculation reproduce sequential recirculation across models and scenarios, including generation after Jacobi prefill. The number of rounds needed grows with the mixing strength. Two rounds suffice for the fixed coefficients used in practice, whereas trained mixers, which mix more strongly, can require three rounds. (3) We analyze the runtime and memory of Jacobi recirculation across models, scenarios, and hardware. It reduces the cost of prefill, evaluation, hyperparameter selection, and mixer training, most strongly when each device processes few sequences at once (Figure 1C).

2 Preliminaries

Decoder-only transformer. Consider a decoder-only transformer (Vaswani et al., 2017; Radford et al., 2019) with L layers and hidden size H that processes tokens $x_0, \dots, x_{T-1}$. We write $z_{\ell,t} \in \mathbb{R}^H$ for the state of token t after layer ℓ, with $z_{-1,t}$ its embedding. Layer ℓ computes $(q_{\ell,t}, k_{\ell,t}, v_{\ell,t}) = \mathrm{QKV}_\ell(z_{\ell-1,t})$ and attends over the keys and values at positions $S_\ell(t) \subseteq \{0, \dots, t\}$, which contain all earlier positions for global attention and the most recent W for sliding-window attention (Beltagy et al., 2020). They form the *KV cache* (Pope et al., 2023; Kwon et al., 2023). The model predicts x_{t+1} from logits $\ell_t = \mathrm{LMHead}(\mathrm{Norm}(z_{L-1,t}))$. When the entire sequence is known, as in prompt prefill, teacher-forced evaluation, and training, all positions are processed in parallel.

Recirculation. Recirculation feeds the state of each token at a deep layer back into an earlier layer, so that the updated KV cache carries deeper representations for later tokens. The pretrained weights remain frozen, and only the mixing rule is tuned. It is specified by a source layer s, a destination layer $d < s$, and a mixing rule $\mathcal{M}_\theta$ with parameters θ. The pair closes a loop through

Algorithm 1 Jacobi recirculation

Require: tokens $x_{0:T-1}$; layers $d < s$; mixing rule $\mathcal{M}_\theta$; rounds $K \geq 1$

1: Run a plain pass through layer s; retain z_d, $z_s^{(0)}$, and the cache $C^{\mathrm{pl}}[0{:}d]$ of layers $0, \ldots, d$
2: **for** $k = 1, \ldots, K$ **do**
3: $\quad r_k \leftarrow s$ if $k < K$, otherwise $L - 1$
4: $\quad \widetilde{z}_d^{(k)} \leftarrow \mathcal{M}_\theta(z_d, z_s^{(k-1)})$
5: $\quad C^{(k)}[d{+}1{:}r_k] \leftarrow \textsc{Recirculate}(\widetilde{z}_d^{(k)})$ ▷ fresh KV at all positions
6: $\quad z_s^{(k)} \leftarrow \textsc{Source}(z_d; C^{(k)}[d{+}1{:}r_k])$ ▷ reads $C^{(k)}$ at earlier positions
7: **end for**
8: **return** the logits $\ell^{(K)}$ of the last source pass and the cache $C^{\mathrm{pl}}[0{:}d] \cup C^{(K)}[d{+}1{:}L{-}1]$

layers $d+1, \ldots, s$, and we call $p = s - d$ its loop length. Once the cache entries of tokens $0, \ldots, t-1$ have been finalized, token t is processed in three steps. (i) *Source pass.* Run the ordinary forward computation of token t against the current cache. This gives its states at every layer and the source logits ℓ_t. (ii) *Mixing.* Compute $\widetilde{z}_{d,t} = \mathcal{M}_\theta(z_{d,t}, z_{s,t})$. Following Mozer et al. (2026), $\mathcal{M}_\theta(z_{d,t}, z_{s,t}) = \beta_\theta \odot z_{d,t} + \alpha_\theta \odot f(z_{s,t})$, where $f(z_{s,t}) = (\|z_{d,t}\|/\|z_{s,t}\|)z_{s,t}$ matches the source norm to the destination norm. The coefficients are either fixed, with the convex rule $\beta_\theta = 1 - \alpha_\theta$ or the additive rule $\beta_\theta = 1$, or produced per dimension by a trained mixer, $(\alpha_\theta, \beta_\theta) = \sigma(\mathrm{MLP}_\theta([z_{s,t}; z_{d,t}]))$, an MLP with two hidden layers followed by an elementwise sigmoid. (iii) *Recirculated pass.* Starting from $\widetilde{z}_{d,t}$, recompute layers $d+1, \ldots, L-1$ against the finalized cache of earlier tokens, and overwrite the keys and values of token t at these layers. The prediction still comes from the source pass, so the recirculated pass affects later tokens only through the cache. Token $t+1$ then reads the updated cache, so tokens are processed one at a time.

3 Jacobi Recirculation

3.1 Recirculation as a fixed-point problem

Let $C_t \in \mathbb{R}^D$ be the cache column written by the recirculated pass of token t, which concatenates the keys and values of layers $d{+}1, \ldots, L{-}1$, each of dimension H_{kv}, so that $D = 2(L{-}d{-}1)H_{\mathrm{kv}}$. Entries at layers $\leq d$ do not depend on mixing and stay fixed. Given a prefix $C_{<t} = (C_0, \ldots, C_{t-1})$, the three steps of Section 2 define a cache update G_t and a source readout R_t, with $C_t = G_t(C_{<t})$ and $\ell_t = R_t(C_{<t})$ for $t = 0, \ldots, T-1$. Writing $C = (C_0, \ldots, C_{T-1})$ and $G(C) = (G_t(C_{<t}))_{t=0}^{T-1}$ gives the fixed-point equation $C = G(C)$. Since each column depends only on earlier columns, the system is strictly lower triangular and has a unique fixed point $C^\star$, which sequential recirculation (SR) computes by forward substitution, with source logits $\ell^\star = R(C^\star)$.

3.2 Jacobi recirculation

We propose *Jacobi recirculation* (JR), which solves $C = G(C)$ by Jacobi-style iteration (Bertsekas & Tsitsiklis, 2015) and updates all positions in parallel in each round. Write $z_d = (z_{d,t})_{t=0}^{T-1}$ for the fixed destination states and $z_s^{(k)} = (z_{s,t}^{(k)})_{t=0}^{T-1}$ for the source states after round k. An ordinary forward pass provides z_d and the initial source states $z_s^{(0)}$, and its cache initializes $C^{(0)} = C^{\mathrm{pl}}$. In round k, we first mix z_d with $z_s^{(k-1)}$ at every position. A recirculated pass then computes and attends to fresh keys and values, producing the new cache $C^{(k)}$. A source pass reads $C^{(k)}$ and produces $z_s^{(k)}$ for the next round. We denote the resulting cache-to-cache map by $\widehat{G}_\theta$, so that $C^{(k)} = \widehat{G}_\theta(C^{(k-1)})$ for $k = 1, \ldots, K$. Intermediate rounds stop at layer s, while the final round continues to layer $L-1$ and returns $\ell^{(K)} = R(C^{(K)})$. Algorithm 1 gives the complete procedure. Jacobi recirculation reproduces sequential recirculation exactly after at most T rounds (Lemma 1), but with fewer rounds its iterates generally deviate from it. Compared with latent-reasoning models trained through a few such rounds, recirculation leaves the pretrained model unchanged. The effect of Jacobi recirculation therefore depends entirely on how well a few rounds approximate the ground truth, a question that prior work has not addressed at scale. We answer it theoretically (Section 3.3) and empirically (Sections 3.4 and 4).

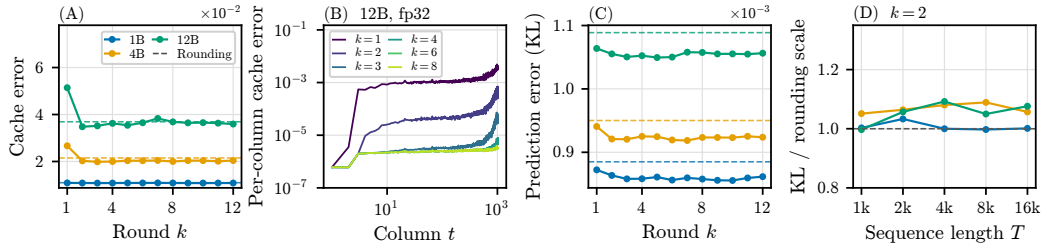


Figure 2: Convergence on Gemma 3 1B, 4B, and 12B. (A) Cache error versus rounds in bf16. (B) Median per-column cache error of 12B after k rounds in fp32, where curves from round k onward overlap on the first k columns (Lemma 1). (C) Prediction error versus rounds in bf16. (D) Prediction error after two rounds, divided by the rounding scale at the same length, versus sequence length in bf16. Dashed lines mark the rounding scale. Figure 6 shows (B) and (D) for all models.

3.3 Convergence guarantees

We establish three guarantees below, which are proved in Appendix B: exactness after finitely many rounds, geometric convergence under local contraction, and convergence of the training gradient.

Lemma 1 (Exactness in finitely many rounds)**.** *For any initialization for which the evaluated passes are well-defined and any $k \geq 1$: (i) $C_t^{(k)} = C_t^\star$ for all $t \leq k-1$; and (ii) the source states and logits of round k are exact for all $t \leq k$. Hence, after K rounds, the returned cache and logits are exact on the first K token positions, and $K = T$ recovers sequential recirculation exactly.*

Lemma 1 needs up to T rounds, whereas we use $K \ll T$. We therefore analyze the finite-round error under local contraction. Let $\|\cdot\|$ be any norm on a cache column and $\|C\|_\infty = \max_t \|C_t\|$.

Proposition 1 (Geometric convergence under a local contraction)**.** *Suppose there are $0 \leq \rho < 1$ and $R \geq \|C^{(0)} - C^\star\|_\infty$ such that $\|\widehat{G}_{\theta,t}(C_{<t}) - \widehat{G}_{\theta,t}(C'_{<t})\| \leq \rho \max_{j<t} \|C_j - C'_j\|$ for every t and every C, C' in the ball $\{C : \|C - C^\star\|_\infty \leq R\}$, with $\max \varnothing := 0$. Then all iterates remain in the ball and $\|C^{(k)} - C^\star\|_\infty \leq \rho^k \|C^{(0)} - C^\star\|_\infty$. If every source readout R_t is L_R-Lipschitz on the same ball, from $\|\cdot\|_\infty$ on caches to the Euclidean norm on logits, then $\max_t \|\ell_t^{(k)} - \ell_t^\star\|_2 \leq L_R \rho^k \|C^{(0)} - C^\star\|_\infty$.*

Note that these bounds are length-independent when ρ, L_R, and the initial error are uniform in T.

A sufficient condition for local contraction. Appendices B.1 and B.4 derive the condition of Proposition 1 from sensitivity bounds on the frozen layers and the mixer. These bounds define the feedback gain $\chi = \varepsilon\lambda[\prod_{i=1}^p (a_i + b_i\kappa_i) - \prod_{i=1}^p a_i]$, where ε is the mixing strength ($\varepsilon = \alpha$ for the fixed rules), a_i, b_i, and κ_i bound the sensitivity of the i-th loop layer, and λ bounds that of the mixer. If $\chi < 1$, the iteration contracts at any rate $\rho \in (\chi, 1)$. Since χ grows with both the mixing strength ε and the number of loop layers p, weaker mixing and shorter loops lead to faster convergence (Section 3.4).

Convergence of the training gradient. We also train the mixer parameters θ through Jacobi recirculation. Let $\mathcal{F}_\star(\theta)$ and $\mathcal{F}_K(\theta)$ denote the prediction losses of sequential recirculation and of K Jacobi rounds. If the iterates stay in a locally contractive neighborhood with factor $q < 1$, Appendix B.2 proves under smoothness assumptions that $\|\nabla_\theta \mathcal{F}_K - \nabla_\theta \mathcal{F}_\star\|_2 \leq A_g q^K + B_g K q^{K-1}$, with constants A_g and B_g independent of K. The first term stems from the forward error of the cache, and the second from evaluating the Jacobians at inexact iterates. Both vanish as K increases.

3.4 Empirical convergence

We now test these results on pretrained models. Unless stated otherwise, we use Gemma 3 1B, 4B, and 12B (Gemma Team et al., 2025) in bf16 on 64 windows of 1,024 tokens from arXiv test documents (Cohan et al., 2018), with the layer pairs and mixing coefficients of the perplexity and arXiv-mixer experiments of Section 4.1. Appendix C gives the experimental details, and Appendix D gives the results for every model and precision and further analyses.

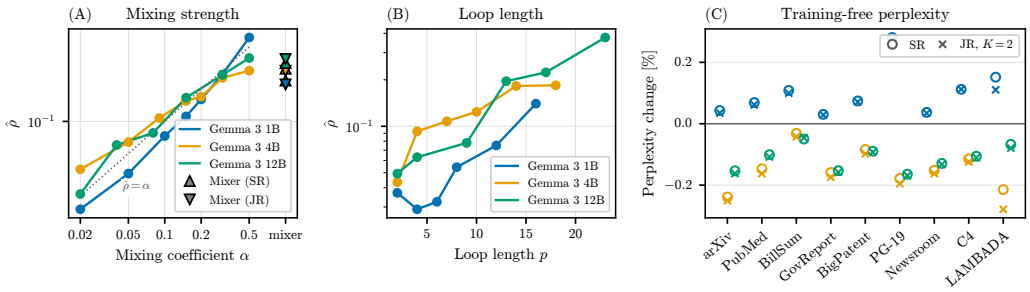


Figure 3: (A, B) Per-round contraction $\hat{\rho}$, the median ratio of the cache errors after successive rounds, versus (A) the mixing coefficient α, with the arXiv mixers trained through sequential (▲) or Jacobi (▼) recirculation, and (B) the loop length p. (C) Perplexity change in % caused by training-free recirculation on nine datasets, with JR using $K{=}2$. All panels use the settings of Section 4.1.

Table 1: Training-free recirculation on the instruction task, accuracy in %. Base is the model without recirculation. JR uses K rounds. Agreement is the fraction of identical SR and JR answers, with the rounding scale in parentheses.

Model	Words	Base ↑	SR ↑	JR $K{=}1$ / 2 ↑	Agreement $K{=}1$ / 2 ↑
4B-IT $(18, 8)$, $\alpha{=}0.15$	Original	82.9	85.6	79.6 / 86.0	93.2 / 97.9 (98.0)
4B-IT $(18, 8)$, $\alpha{=}0.15$	Held-out	79.8	93.2	86.1 / 92.2	92.6 / 97.8 (96.9)
12B-IT $(29, 16)$, $\alpha{=}0.10$	Original	80.1	100.0	99.8 / 100.0	99.8 / 100.0 (100.0)
12B-IT $(29, 16)$, $\alpha{=}0.10$	Held-out	94.2	100.0	100.0 / 100.0	100.0 / 100.0 (100.0)

Cache convergence. Let $C^\star$ be the cache of sequential recirculation at the same precision and $C^{(k)}$ the cache after k rounds. We measure the relative error of cache column t, $e_t^{(k)} = \|C_t^{(k)} - C_t^\star\|_2 / \|C_t^\star\|_2$, and the overall cache error $e_{\text{cache}}^{(k)} = \|C^{(k)} - C^\star\|_F / \|C^\star\|_F$. We compare these errors with the *rounding scale*, estimated from two runs of sequential recirculation with batch sizes 4 and 1, to remove the rounding variation across runs. The overall cache error reaches the rounding scale within two rounds (Figure 2(A)). In fp32, which resolves the iteration below this scale, all curves from round k onward overlap on the first k cache columns, as predicted by Lemma 1 (Figure 2(B)). In general, each round reduces the median error to between 3% and 16% of its previous value, until it levels off near the fp32 rounding scale. We also measure the prediction error as the average KL divergence $e_{\text{pred}}^{(k)} = (T-1)^{-1} \sum_{t=0}^{T-2} D_{\text{KL}}(p_t^\star \| p_t^{(k)})$ between $p_t^\star = \text{softmax}(\ell_t^\star)$ and $p_t^{(k)} = \text{softmax}(\ell_t^{(k)})$, with its rounding scale computed likewise. After one round, the prediction error of all models already reaches the rounding scale (Figure 2(C)).

Sequence length. Proposition 1 gives a length-independent rate when its constants are uniform over lengths. On prefixes of 16 PG-19 documents (Rae et al., 2020) from 1,024 to 16,384 tokens, the prediction error after two rounds stays within 10% of the rounding scale for all models and lengths (Figure 2(D)), so in bf16 the number of rounds needed does not grow with the sequence length.

Mixing strength and loop length. Section 3.3 predicts faster convergence for weaker mixing and shorter loops. We test this with the per-round contraction $\hat{\rho}$, which estimates ρ in Proposition 1. We vary either the mixing coefficient or the loop length around the settings of Section 4.1 in fp32. As in Figure 3, we find that both predictions hold. $\hat{\rho}$ grows by an order of magnitude as α increases from 0.02 to 0.5, and severalfold as the loop length increases from 2 to 16–23 layers. Note that longer loops also start further from the exact cache. After one round, their cache error is one to two orders of magnitude larger, and their prediction error is two to three orders of magnitude larger.

4 Experiments

Setup. We use Gemma 3 1B, 4B, and 12B, with the instruction-tuned variants 4B-IT and 12B-IT for the instruction task, and also evaluate Llama 3.1 8B (Grattafiori et al., 2024) and Qwen3 8B (Yang et al., 2025). Computation is in bf16, and Jacobi recirculation uses $K = 2$ unless stated otherwise.

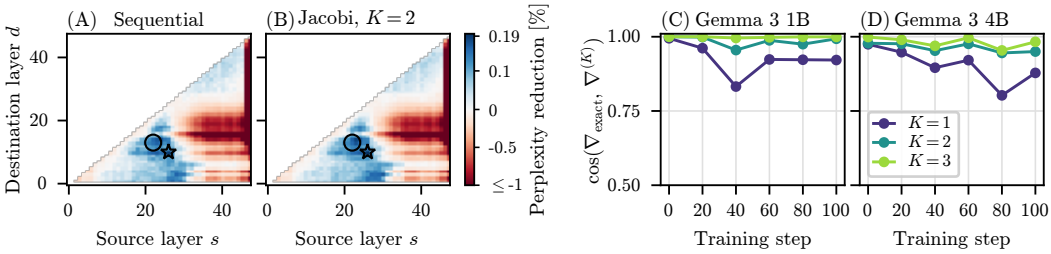


Figure 4: (A, B) Training-free perplexity reduction on every layer pair of Gemma 3 12B at the mixing coefficient of Mozer et al. (2026), with their pair circled and the best pair starred. (C, D) Mean cosine between the exact and the K-round gradient in fp32 on the same eight training windows, at checkpoints along the training of the arXiv mixer through Jacobi recirculation.

Table 2: Mixers trained through SR or JR with $K{=}2$, mean $\pm$ standard deviation over three seeds. Base is the model without recirculation. Left: arXiv perplexity, with each mixer evaluated through its training method. Right: GSM8K pass@1 in %, with all mixers evaluated through SR.

arXiv perplexity				GSM8K pass@1			
Model	Base ↓	SR ↓	JR ↓	Model	Base ↑	SR ↑	JR ↑
1B	11.406	10.785±0.015	10.792±0.037	4B $(16, 5)$	31.0	38.2±1.3	40.3±1.6
4B	7.809	7.476±0.022	7.463±0.003	12B $(35, 16)$	43.9	67.7±1.4	66.7±1.2
12B	6.693	6.480±0.076	6.454±0.020	Llama 8B $(13, 5)$	43.3	48.8±0.8	47.5±0.9
				Qwen 8B $(13, 5)$	81.4	83.0±0.6	82.6±0.2

For perplexity, we use eight long-document datasets, arXiv and PubMed, GovReport (Huang et al., 2021), BillSum (Kornilova & Eidelman, 2019), BigPatent (Sharma et al., 2019), PG-19, Newsroom (Grusky et al., 2018), and C4 (Raffel et al., 2020), together with LAMBADA (Paperno et al., 2016). All evaluations except the layer-pair and coefficient sweeps use complete test sets, apart from BigPatent and C4, of which we use the first 10,000 test and 25,000 validation documents, and SVAMP, for which we pool its training and test splits. As in Section 3.4, differences comparable to the rounding scale indicate that the two computations agree up to numerical precision. Mixers follow the architecture and training recipe of Mozer et al. (2026). Appendix C gives the experimental details, and Appendix E gives further results and analyses.

4.1 Exactness

We now test whether the convergence of Section 3.4 carries over to the outcome of each scenario.

Instruction following. We first consider the instruction task of Mozer et al. (2026), on which training-free recirculation helps substantially. Each question presents a fruit and an animal and asks for the position of one of them, read from candidate-token logits. Following their description, we built the 800 questions and used their layer pairs for Gemma 3 4B-IT and 12B-IT and a held-out word list. As shown in Table 1, we find that two rounds suffice to reproduce the sequential accuracy within one point, and that Jacobi recirculation and sequential recirculation give the same answer on 98–100% of questions, matching the rounding scale. One round suffices on 12B-IT but not on 4B-IT. On Llama 3.1 8B and Qwen3 8B, two rounds also match sequential recirculation (Appendix E).

Perplexity and hyperparameter selection. Sequential recirculation is found to be sensitive to the hyperparameters. Mozer et al. (2026) select the layer pair by training-free perplexity and reuse it on downstream tasks. At the configurations they selected, we find that the two computations differ in perplexity by at most 0.02% on eight datasets and by less than 0.1% on LAMBADA (Figure 3(C)). Both of them show a slight improvement across all datasets for 4B and 12B, and a slight degradation on 1B. We then test whether using Jacobi recirculation for hyperparameter selection leads to the same choices by evaluating every layer pair with $s > d \geq 1$ on 128 arXiv training windows, 300 to 1,081 pairs per model. As shown in Figure 4(A, B), the two computations yield nearly identical perplexity landscapes, with Spearman rank correlations of at least 0.977 across the five models. Jacobi recirculation selects the same best pair as sequential recirculation on all three Gemma

Table 3: Pass@1 in % on GSM8K and SVAMP, mean ± standard deviation over three seeds, with the GSM8K mixers trained through Jacobi recirculation and applied to SVAMP without further training. JR computes only the prompt cache, and the answer is decoded greedily with sequential recirculation. With three rounds, every difference between SR and JR is within seed variation.

	GSM8K				SVAMP			
Model	Base	SR	JR $K{=}2$	JR $K{=}3$	Base	SR	JR $K{=}2$	JR $K{=}3$
4B $(16, 5)$	31.0	40.3±1.6	38.5±1.1	40.0±1.5	50.9	55.0±2.4	52.7±2.2	54.6±1.3
12B $(35, 16)$	43.9	66.7±1.2	66.5±1.5	66.7±0.8	52.0	65.6±3.4	63.8±3.3	64.9±2.9
Llama 3.1 8B $(13, 5)$	43.3	47.5±0.9	47.8±0.5	47.5±0.1	49.3	54.3±2.5	54.8±1.5	54.6±2.3
Qwen3 8B $(13, 5)$	81.4	82.6±0.2	82.6±0.4	82.6±0.6	84.0	84.2±1.4	83.9±1.5	84.5±1.4

models, and on Llama 3.1 8B and Qwen3 8B a nearby pair within 0.02 percentage points of the best. Sweeping α at fixed pairs, it selects the same best α in 7 of 9 sweeps and an adjacent value in the other two.

Mixer training. We further ask whether mixers trained through Jacobi recirculation perform as well as those trained through sequential recirculation. We first train arXiv mixers on each computation and then evaluate each mixer using its training method. We find that the two reach the same perplexity across seeds (Table 2, left). To isolate the effect of training, we then train GSM8K mixers on both computations and evaluate them with sequential recirculation (Table 2, right). The mixers trained through Jacobi recirculation reach a similar pass@1 across all models, with differences comparable to the variation across seeds. This suggests that the gradient through two rounds is close to the exact gradient. To verify this, we compute both gradients in fp32 on the same eight training windows at checkpoints along the training trajectory. We observe that with two rounds, their mean cosine stays above 0.94 at every checkpoint (Figure 4(C, D)), consistent with our theory in Section 3.3. We also find that a few windows show lower values, which did not affect the trained mixers.

Prefill for generation. In generation tasks, Jacobi recirculation computes only the prompt cache, and the answer is then decoded token by token with sequential recirculation. Note that small differences in the prompt cache can propagate into the generated text, making generation a harder test than scoring. We study this on GSM8K (Cobbe et al., 2021) with mixers trained through Jacobi recirculation. As shown in Table 3, we find that with two rounds, Jacobi recirculation still matches sequential recirculation on 12B, Llama 3.1 8B, and Qwen3 8B, but is 1.8 points below on 4B. Since the fixed point does not depend on the number of rounds, rounds can be added at test time. With three rounds, the gap on 4B shrinks to 0.3 points. However, we find that the generated answers can differ more than the accuracy difference. With two rounds, the final answers of the two computations agree on 58% to 90% of the questions, compared with 86% to 94% at the rounding scale, and additional rounds narrow this gap in most cases. We also find that the mixers trained through Jacobi recirculation can transfer directly to other tasks. On SVAMP (Patel et al., 2021), it improves pass@1 except on Qwen3 8B, whose base accuracy is already high. With two rounds, Jacobi recirculation matches sequential recirculation on Llama 3.1 8B and Qwen3 8B, but falls about 2 points below it on both Gemma models. With three rounds, the difference is below 1 point on every model.

Across these scenarios, the number of rounds needed grows with the mixing strength and loop length, as Section 3.4 predicts. For fixed mixing coefficients, two rounds deviate noticeably only at configurations that make perplexity worse, which are not used in practice (Figure 12), while trained mixers mix more strongly (Figure 3(A)) and sometimes require three rounds in greedy generation.

4.2 Efficiency

Jacobi recirculation trades additional computation for parallelism over the sequence. To analyze its efficiency, we model this trade-off for prefill, which all our scenarios share, test the model across sequence lengths, batch sizes, and numbers of rounds, and then time each scenario end to end. Appendix F gives further details and results, including results on NVIDIA A100 and H100 GPUs.

Computation and memory. We count one unit for processing all T tokens through one layer, whether together or separately. The three methods require $E_{\text{plain}} = L$, $E_{\text{seq}} = (d{+}1){+}2(L{-}d{-}1)$, and $E_{\text{Jac}} = (s+1) + 2(K-1)p + 2(L-d-1)$ layer evaluations, respectively. All three compute layers $0, \ldots, d$ once. Jacobi recirculation additionally initializes the source states through layer s,

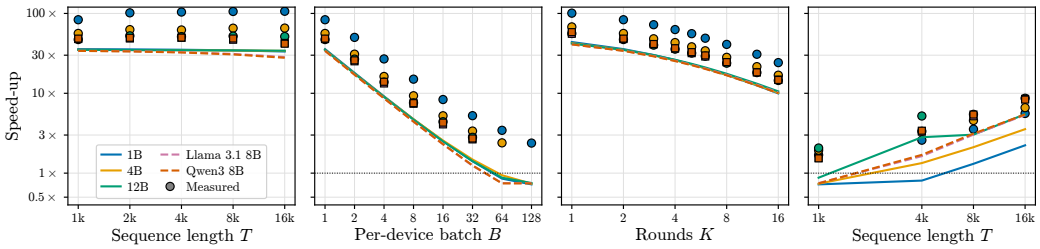


Figure 5: Prefill speed-up by Jacobi recirculation on one MI210, excluding the LM head, for Gemma 3 (solid) and Llama 3.1 8B and Qwen3 8B (dashed). The first three panels vary T, B, or K from a single sequence of $T = 1{,}024$ tokens with $K = 2$, and the right panel varies T at the largest batch that fits sequential recirculation. Markers show measured values and lines show S_{ideal}.

executes two passes through layers $d+1, \ldots, s$ in each intermediate round, and runs both passes to the top in the final round. Each additional round, therefore, costs $2p$ layer evaluations. Both methods produce the same KV cache, which, together with the weights, dominates their memory usage; thus, their peak memory usage during prefill is similar. In training, backpropagation stores the evaluation of every layer, so the memory required by Jacobi training grows with E_{Jac}.

Ideal speed-up. Note that runtime depends not only on the amount of computation but also on memory traffic. For a single sequence, sequential recirculation batches the recirculated pass of token $t-1$ with the source pass of token t, so each weight load serves only two token computations per step, whereas Jacobi recirculation processes all T positions in each pass. In single-sequence prefill, sequential recirculation is therefore memory-bound and Jacobi recirculation compute-bound (Pope et al., 2023). The roofline model (Williams et al., 2009) gives a runtime lower bound of $\max(F/P, Q/\beta_{\text{mem}})$ for F floating-point operations and Q bytes of memory traffic on a device with peak compute throughput P and memory bandwidth β_{mem}. The ratio of the bounds for the two methods defines the ideal speed-up $S_{\text{ideal}} := \max(F_{\text{seq}}/P,\, Q_{\text{seq}}/\beta_{\text{mem}}) / \max(F_{\text{Jac}}/P,\, Q_{\text{Jac}}/\beta_{\text{mem}})$. In this regime, $S_{\text{ideal}} = (P/\beta_{\text{mem}})(Q_{\text{seq}}/F_{\text{Jac}})$. For B sequences per device and comparable layer sizes, neglecting attention and activation traffic gives the scaling $S_{\text{ideal}} \propto (P/\beta_{\text{mem}})(L-d-1)/(BE_{\text{Jac}})$. The sequence length cancels, because the weight traffic of sequential recirculation and the arithmetic of Jacobi recirculation both grow linearly with T. The speed-up instead decreases with the per-device batch B, which adds arithmetic to Jacobi recirculation but no weight traffic to sequential recirculation, and with the number of rounds K, which increases E_{Jac}. In practice, device memory limits B, so longer sequences and larger models raise the speed-up at the largest batch that fits. On an AMD MI210, the ideal speed-up is about $35\times$ for all three Gemma 3 models at $T = 1{,}024$, $B = 1$, and $K = 2$, and it is higher on GPUs with a higher ratio P/β_{mem}. Appendix F.3 reports measurements on NVIDIA A100 and H100 GPUs.

Prefill time. We time prefill on one MI210 on 16 long PG-19 documents, with both implementations compiled and captured in CUDA graphs. Unless stated otherwise, $K = 2$ and timings exclude the LM head. At $T = 1{,}024$ and $B = 1$, Jacobi recirculation is 47–83$\times$ faster than sequential recirculation on the three Gemma 3 models and $48\times$ on Llama 3.1 8B and Qwen3 8B, while using 1.3–1.4$\times$ the FLOPs and a similar amount of peak memory. Compared with a plain forward pass, Jacobi recirculation is 2.2–2.4$\times$ slower, matching its ratio of layer evaluations, whereas sequential recirculation is 105–203$\times$ slower. Notably, these speed-ups exceed the ideal estimate analyzed above. We find that this is because the two computations fall short of their hardware limits by different amounts. The ideal estimate assumes that sequential recirculation reaches the peak bandwidth and Jacobi recirculation the peak compute, so the measured speed-up equals S_{ideal} scaled by $\text{MFU}_{\text{Jac}}/\text{MBU}_{\text{seq}}$, where the model FLOP utilization $\text{MFU} = F/(Pt)$ and the memory bandwidth utilization $\text{MBU} = Q/(\beta_{\text{mem}}t)$ are the attained fractions of these peaks. Empirically, Jacobi recirculation attains 38–60% of peak compute, whereas sequential recirculation attains only 16% of peak bandwidth on 1B, rising to 41–43% on the 8B and 12B models. The low-bandwidth use of sequential recirculation thus explains why the measured speed-up exceeds the ideal and why it shrinks toward the ideal estimate as the model size increases. In isolation, we find the matrix products of sequential recirculation reach only 23–51% of peak bandwidth when processing two tokens at a time (Appendix F.2), so we conjecture that kernels tailored to such small inputs could close this gap.

Table 4: End-to-end running time with SR and JR ($K=2$). Base is the model without recirculation, Batch counts sequences per device, and FLOPs JR/SR is their computation ratio. Perplexity times are GPU-hours summed over all devices, including the LM head at every position. All other times are wall-clock times on one device, or on eight devices for training. Training times exclude the first of 100 steps of 32 windows, which contains compilation. On GSM8K, Base is omitted because it generates longer answers without a trained mixer, and parentheses give the prompt-phase speed-up.

Scenario	Model	Batch	Base	SR	JR	Speed-up	FLOPs JR/SR
Perplexity, 8 datasets	1B	32	2.8 h	19.0 h	4.8 h	4.0×	1.31
	4B	32	7.3 h	44.9 h	14.3 h	3.1×	1.32
	12B	32	19.7 h	110 h	40.3 h	2.7×	1.31
Instruction task	4B-IT	64	4.5 s	18.4 s	12.2 s	1.5×	1.51
	12B-IT	64	13.7 s	44.0 s	35.6 s	1.2×	1.49
GSM8K	1B	64	—	1,146 s	1,055 s	1.1× (8.6×)	1.09
	4B	64	—	1,255 s	1,117 s	1.1× (5.8×)	1.18
	12B	64	—	1,996 s	1,737 s	1.1× (4.1×)	1.26
Training, arXiv	1B	4	—	26.4 min	1.3 min	20×	1.31
	4B	4	—	1.07 h	3.2 min	20×	1.32
	12B	2	—	3.84 h	8.2 min	28×	1.31

Scaling. We further test each factor in the scaling of S_{ideal} (Figure 5). As predicted, the measured speed-up remains roughly constant with the sequence length at a fixed batch and decreases with the per-device batch, while remaining above the ideal value. Since only the per-device batch matters, data parallelism over eight devices preserves the single-device speed-up. With the device memory fixed, longer contexts reduce the largest batch that fits within sequential recirculation, from 64–512 sequences at 1k tokens to 8–32 at 16k tokens, thereby raising the speed-up from 1.5–$2.1\times$ to 5.6–$8.7\times$. It grows fastest on Llama 3.1 8B and Qwen3 8B, which use no sliding-window attention and thus have larger caches. The benefit therefore grows with the context length, which suits long-context scenarios. Finally, the layer pair and the number of rounds enter S_{ideal} through the factor $(L-d-1)/E_{\text{Jac}}$. As shown in Appendix F.2, we find that the measured speed-up follows this factor with log-scale correlations of 0.94 to 0.99 across the layer pairs of Section 4.1.

Complete evaluations. Next, we time the complete evaluations in Section 4.1. Results are shown in Table 4, with more in Appendix F.2. Unlike a single prompt, up to 64 sequences per device are used there, limiting the speed-up. Perplexity and the instruction task, which involve no decoding, run 1.2–$4.0\times$ faster with Jacobi recirculation. Compared with the model without recirculation, sequential recirculation is 3.2–$6.7\times$ slower, while Jacobi recirculation is only 1.7–$2.7\times$ slower. On GSM8K, where token-by-token decoding dominates the runtime and remains unchanged, Jacobi recirculation shortens the prompt phase by 4–$9\times$ but the total runtime only by 1.1–$1.3\times$. Jacobi recirculation also accelerates the layer-pair sweeps in Section 4.1 by $3\times$. Jacobi recirculation can thus replace sequential recirculation in all these scenarios, making recirculation cheaper to study and to use.

Mixer training. Training shows a much larger gain, as its per-device batch is smaller. With the training setups of Section 4.1, Jacobi recirculation trains the arXiv mixers 20–$28\times$ faster and the GSM8K mixers 14–$17\times$ faster (Tables 4 and 27). There, each device processes only one to four sequences at a time, which keeps sequential recirculation memory-bound. Even when each method uses the largest batch that fits on a single device, Jacobi training remains $11.8\times$ faster on 1B and $17.5\times$ faster on 4B. On the other hand, Jacobi training requires 1.07 to 1.26 times the peak memory of sequential training at the same batch, and each further round adds 15% to 20%.

5 Conclusion

We propose Jacobi recirculation, which replaces the token-wise recursion of recirculation with a few rounds of parallel updates. The iteration is exact after finitely many rounds and converges faster with weaker mixing and shorter loops. In our experiments, two to three rounds suffice to reproduce the performance of sequential recirculation. The speed-up is largest when few sequences share a device, as in single-request prefill, mixer training, and long contexts. Overall, our results make recirculation much cheaper to use and to study. We discuss the limitations and future work in Appendix A.

AI USE STATEMENT

We used generative AI tools (Claude and Codex) as assistants throughout this project, and most of the work was done jointly by the authors and these tools. For tasks with required disclosure, the tools assisted in developing the theoretical framework of Section 3, including its statements, and in writing and checking the proofs of Appendix B. The authors had the initial idea, which took its final form in discussions with these tools. The tools helped design the experiments, mainly the efficiency analysis of Section 4.2, and gave feedback on the methodology. They also assisted in implementing the methods, preparing and reformatting the datasets, and interpreting the results. We did not use generative AI tools to generate synthetic datasets, and translation and qualitative or thematic data analysis are not applicable to this work. For tasks with recommended disclosure, we used these tools to suggest experimental parameters, to create and edit code, figures, and the released code, to draft and edit parts of the paper and suggest its structure, for brainstorming, and to search, summarize, and format the literature and references. We have reviewed all AI-assisted work. The authors checked the proofs, validated the implementations against reference implementations and reruns (Appendix C), and verified each reference against its official publication record. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Ahmad Ajalloeian and Sebastian U. Stich. On the convergence of SGD with biased gradients. *arXiv preprint arXiv:2008.00051*, 2020. URL https://arxiv.org/abs/2008.00051.

Peter L. Bartlett, Dylan J. Foster, and Matus J. Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Dimitri Bertsekas and John Tsitsiklis. *Parallel and distributed computation: numerical methods*. Athena Scientific, 2015.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Ziyang Cai, Xingyu Zhu, Yihe Dong, Yinghui He, and Sanjeev Arora. T^2MLR: Transformer with temporal middle-layer recurrence. *arXiv preprint arXiv:2607.15178*, 2026. URL https://arxiv.org/abs/2607.15178.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 2 (short papers)*, pp. 615–621, 2018.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16344–16359, 2022.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Angela Fan, Thibaut Lavril, Edouard Grave, Armand Joulin, and Sainbayar Sukhbaatar. Addressing some limitations of transformers with feedback memory. *arXiv preprint arXiv:2002.09402*, 2020.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of LLM inference using lookahead decoding. In *International Conference on Machine Learning*, 2024.

Jonas Geiping, Sean Michael McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=S3GhJooWIC.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013. doi: 10.1137/120880811.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pp. 11398–11442. PMLR, 2023.

Xavier Gonzalez, Andrew Warrington, Jimmy T. H. Smith, and Scott W. Linderman. Towards scalable and stable parallelization of nonlinear RNNs. In *Advances in Neural Information Processing Systems*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Riccardo Grazzi, Luca Franceschi, Massimiliano Pontil, and Saverio Salzo. On the iteration complexity of hypergradient computation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3748–3758, 2020. URL https://proceedings.mlr.press/v119/grazzi20a.html.

Max Grusky, Mor Naaman, and Yoav Artzi. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 708–719, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1065. URL https://aclanthology.org/N18-1065/.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Itxz7S4Ip3.

Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient attentions for long document summarization. In *Proceedings of the 2021 conference of the north American chapter of the association for computational linguistics: Human language technologies*, pp. 1419–1436, 2021.

Zeyi Huang, Xuehai He, LiLiang Ren, Yiping Wang, Baolin Peng, Hao Cheng, Shuohang Wang, Pengcheng He, Jianfeng Gao, Yong Jae Lee, and Yelong Shen. Latent recurrent transformer: Architecture exploration, training strategies, and scaling behavior. *arXiv preprint arXiv:2605.26797*, 2026. URL https://arxiv.org/abs/2605.26797.

Anastassia Kornilova and Vladimir Eidelman. BillSum: A corpus for automatic summarization of US legislation. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pp. 48–56, 2019.

Siqi Kou, Lanxiang Hu, Zhezhi He, Zhijie Deng, and Hao Zhang. CLLMs: Consistency large language models. In *International Conference on Machine Learning*, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Yi Heng Lim, Qi Zhu, Joshua Selfridge, and Muhammad Firmansyah Kasim. Parallelizing nonlinear sequential models over the sequence length. In *International Conference on Learning Representations*, 2024.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 1540–1552, 2020.

Michael C. Mozer, Shoaib Ahmed Siddiqui, Danny Sawyer, Sunny Sanyal, and Rosanne Liu. Recirculation. *arXiv preprint arXiv:2608.17981*, 2026. URL https://arxiv.org/abs/2608.17981.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 2080–2094, 2021.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624, 2023.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodolà. Accelerating transformer inference for translation via parallel decoding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=din0lGfZFd.

Eva Sharma, Chen Li, and Lu Wang. BIGPATENT: A large-scale dataset for abstractive and coherent summarization. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 2204–2213, 2019.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 10131–10165, 2025.

Yang Song, Chenlin Meng, Renjie Liao, and Stefano Ermon. Accelerating feedforward computation via parallel nonlinear equation solving. In *International Conference on Machine Learning*, 2021.

Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-Margin training: Scalable certification of perturbation invariance for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

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.

Xi Wang, Ziyang Cai, Zheng Zhan, Harry Dong, Ying Fan, Gustavo de Rosa, Tim Pearce, and John Langford. Full-bandwidth transformer. *arXiv preprint arXiv:2608.08888*, 2026. URL `https://arxiv.org/abs/2608.08888`.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

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.

Haoyi Wu, Zhihao Teng, and Kewei Tu. Parallel continuous chain-of-thought with Jacobi iteration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 914–926, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Boyi Zeng, He Li, Shixiang Song, Yixuan Wang, Zitong Wang, Ziwei He, Xinbing Wang, and Zhouhan Lin. PonderLM-2: Pretraining LLM with latent thoughts in continuous space. In *International Conference on Machine Learning*, 2026.

Contents of the appendix

A Limitations and future work 15
The limitations of our work and directions for future work.

B Statements and proofs 15
The theoretical results summarized in Section 3.3: exactness after finitely many rounds, a sufficient condition for the local contraction of the cache iteration in terms of layerwise sensitivity bounds, the convergence of the training gradient, and a stationarity guarantee for training the mixer with SGD through a finite number of rounds.

B.1 Local contraction from layer and mixer sensitivity bounds 15
B.2 Accuracy of the unrolled training gradient 22
B.3 Training the mixer with SGD 27
B.4 Sharper convergence bounds 28

C Experimental setup 30
The experimental details of Sections 3.4 and 4.

C.1 Models, precision, and hardware 30
C.2 Implementations 31
C.3 Recirculation settings 31
C.4 Data and tasks 33
C.5 Metrics and rounding scales 35
C.6 Mixer training 35
C.7 Timing protocol 36

D Convergence: supplementary results 37
Supplements Section 3.4 with results for each model and in the complementary precision, the dependence on the sequence length and the position, the contraction estimates, and a comparison of the rounding scale with the difference between implementations.

E Exactness: supplementary results 40
Supplements Section 4.1 in its order.

E.1 Instruction task 40
E.2 Perplexity 40
E.3 Hyperparameter selection 41
E.4 Mixer training 43
E.5 Generation 43

F Efficiency: supplementary results 44
Supplements Section 4.2 with the derivation of the cost model, the complete timing and memory measurements, and prefill measurements on NVIDIA A100 and H100 GPUs.

F.1 Cost model 44
F.2 Measurements 47
F.3 Other GPUs 52

A LIMITATIONS AND FUTURE WORK

Limitations. We study the form of recirculation proposed by Mozer et al. (2026) and do not consider more general forms. Recirculation is itself recent, and how much and where it helps is not yet settled. Faster computation makes such studies cheaper, which is one motivation of this work. In some scenarios, the speed-up is small, for example when decoding dominates generation or when many sequences share a device (Section 4.2), and optimizations specific to sequential recirculation could narrow the gap further. Jacobi recirculation also adds computation, 1.3–$1.4\times$ the FLOPs of sequential recirculation in prefill, and Jacobi training needs 1.07–$1.26\times$ the memory of exact training. Beyond the cost model, our efficiency results depend on the hardware and the implementation. We therefore time the fastest implementation of each computation on three GPUs and report all settings (Appendix F). Finally, our theory is simple. Its sufficient conditions rest on sensitivity bounds that we do not verify for the models we study, and the experiments test the dependences it predicts rather than the bounds themselves.

Future work. Natural extensions include accelerating more general forms of recirculation, further optimizing both computations, and evaluating them in more practical scenarios, such as serving many concurrent requests. The same analysis, which measures how closely a few parallel rounds approximate the ground truth and relates their cost to the hardware, can be applied to other feedback architectures. With faster computation, recirculation can also be compared at scale and at matched compute with other ways of adding computation to a pretrained model, such as looped transformers and trained latent-reasoning models. Jacobi recirculation itself may also be improved, for example by choosing the number of rounds from the residual between rounds, by solvers faster than Jacobi iteration, or by implicit differentiation for mixer training.

B STATEMENTS AND PROOFS

This appendix proves the results of Section 3.3 and relates them to the experiments. Appendix B.1.1 proves Lemma 1, and Appendix B.1.6 proves Proposition 1. Appendix B.1 derives the condition of Proposition 1 from sensitivity bounds on the frozen layers and the mixer. It shows why the rate of convergence depends on the mixing strength and the loop length, which Figure 3 measures, and when the error bounds do not grow with the sequence length, which Figure 2(D) tests. Appendix B.2 proves the bound on the training gradient of Section 3.3, whose error Figure 4(C, D) measures during mixer training. Appendix B.3 turns this bound into a guarantee for SGD with a fixed number of rounds, the setting of the mixers compared in Table 2. Appendix B.4 sharpens the condition of Appendix B.1 to the feedback gain χ of Section 3.3, under which the error from the plain cache decreases by a factor χ per round.

B.1 LOCAL CONTRACTION FROM LAYER AND MIXER SENSITIVITY BOUNDS

Purpose and scope. Proposition 1 assumes a column-wise contraction inequality for the cache map. Here we derive a sufficient condition from bounds on individual transformer layers and on the source-to-destination mixing rule. Propagation up the frozen layers is triangular in *layer order*. Only the mixing step sends a perturbation from the source layer back to the destination layer. We first bound these two computations separately, then choose layer weights that account for upward amplification, and finally show when the plain cache lies in an invariant contraction neighborhood. All statements concern the full-cache map $\widehat{G}_\theta$ of Section 3.2, whose fixed point solves $C = G(C)$ (Section 3.1), in exact arithmetic, with tokens, parameters and attention masks fixed. Forward computations are deterministic, for example with dropout disabled; if randomness is retained, the same layer maps must be reused in the plain, source and recirculated passes and across rounds.

B.1.1 LAYER NOTATION AND THE LOCAL HYPOTHESES

Fix $d < s$ and put $m := L-d-1$. We index the *affected layers* $d+1, \dots, L-1$ by $i = 1, \dots, m$, so that affected layer i is physical layer $d+i$. The loop of Section 2 consists of affected layers $1, \dots, p$, and affected layer m is the top layer $L-1$. For a cache C and a position t, let $\mathsf{C}_{i,t} \in \mathbb{R}^{2H_{\mathrm{kv}}}$ concatenate the key and the value stored at affected layer i and position t, so that the cache column

of Section 3.1 is $C_t = (\mathsf{C}_{1,t}, \dots, \mathsf{C}_{m,t})$. Write $\mathsf{C}_i := (\mathsf{C}_{i,t})_{t=0}^{T-1}$. This groups the same cache C by layer rather than token.

Equip each per-token block with the Euclidean norm and define

$$\|\mathsf{C}_i\|_{\mathrm{tok}} = \max_{0\le t<T} \|\mathsf{C}_{i,t}\|_2, \qquad \|C\|_{\mathrm{blk}} = \max_{1\le i\le m} \|\mathsf{C}_i\|_{\mathrm{tok}}. \tag{1}$$

The same token-maximum convention is used for arrays of residual-stream states. No sum over token positions appears in these norms. With the column norm $\max_i \|\mathsf{C}_{i,t}\|_2$, $\|C\|_{\mathrm{blk}}$ is the norm $\|C\|_\infty$ of Section 3.3.

Let $F_i(h, \mathsf{C}_i)$ be affected layer i applied to the array h of its input states. At position t, its query and its own key and value are computed from h_t, and the keys and values of earlier positions are read from C_i. Let $V_i(h)$ be the corresponding array of newly computed keys and values, before the attention update. F_i includes the normalization, attention, residual connections and feedforward computation, whereas V_i is the cache extractor. Fix the destination states z_d of Section 3.2, and write the mixing rule of Section 2 as $\mathcal{M}_\theta(z_d, z_s) = z_d + \varepsilon H_\theta(z_d, z_s)$, where $\varepsilon \ge 0$ is the mixing strength. The cache map $\widehat{G}_\theta$ of Section 3.2 depends on ε, and quantities that vary with ε, such as its fixed point $C^\star_\varepsilon$, carry it as a subscript. The source map on a supplied cache and the subsequent recirculated map are then

$$\begin{aligned} S_0(C) &= z_d, & S_i(C) &= F_i(S_{i-1}(C), \mathsf{C}_i), \quad 1 \le i \le m, \\ U_0 &= z_d + \varepsilon H_\theta(z_d, S_p(C)), & & \\ \mathsf{C}_i^+ &= V_i(U_{i-1}), & U_i &= F_i(U_{i-1}, \mathsf{C}_i^+), \qquad 1 \le i \le m, \\ [\widehat{G}_\theta(C)]_i &= \mathsf{C}_i^+. & & \end{aligned} \tag{2}$$

Each recirculated layer thus evaluates $F_i(U_{i-1}, V_i(U_{i-1}))$, so its attention reads the fresh cache computed from all current layer inputs. The source map instead reads the supplied cache at earlier positions and computes only the key and value of the current position. Define $N_{\varepsilon,\theta}(z)$ by replacing $S_p(C)$ with z in the recirculated equations. Then $\widehat{G}_\theta = N_{\varepsilon,\theta} \circ S_p$. After forming $C^{(k)}$, the source pass of round k evaluates $S_p(C^{(k)})$, or $S_m(C^{(k)})$ and the head in the final round. Intermediate rounds of Algorithm 1 stop at layer s. Since $S_p(C)$ depends only on $\mathsf{C}_1, \dots, \mathsf{C}_p$, and the first p layers of $N_{\varepsilon,\theta}(z)$ use no higher layer, every source state is unchanged and the returned cache equals $\widehat{G}_\theta^K(C^{\mathrm{pl}})$. For $k < K$, $C^{(k)}$ denotes the full cache $\widehat{G}_\theta^k(C^{\mathrm{pl}})$, whose layers above p are not computed.

The form $z_d + \varepsilon H_\theta(z_d, z_s)$ includes the fixed convex rule, with $\alpha_\theta = \varepsilon$, $\beta_\theta = 1 - \varepsilon$ and $H_\theta(z_d, z_s) = f(z_s) - z_d$, and the fixed additive rule, with $\alpha_\theta = \varepsilon$, $\beta_\theta = 1$ and $H_\theta(z_d, z_s) = f(z_s)$. For trained mixers, we consider the adaptive family

$$\alpha_\theta = \varepsilon a_\theta(z_d, z_s), \qquad \beta_\theta = 1 + \varepsilon b_\theta(z_d, z_s), \qquad H_\theta = b_\theta \odot z_d + a_\theta \odot f(z_s). \tag{3}$$

Every mixer of Section 2 can be written in this form with $a_\theta = \alpha_\theta/\varepsilon$ and $b_\theta = (\beta_\theta - 1)/\varepsilon$. The results below require these gates to be bounded and Lipschitz, which holds with a small ε only when α_θ is small and β_θ is close to one. In particular, a small value of α_θ does not control source-dependent variation in β_θ or in either gate.

Causality, fixed-point equivalence and exact prefixes. At position j, $S_{p,j}(C)$ depends only on $C_{<j}$. A fresh causal recirculated pass at position t depends only on mixed inputs at positions $j \le t$. Their cache arguments all belong to $C_{<t}$. Consequently $\widehat{G}_{\theta,t}$ depends only on $C_{<t}$, and $\widehat{G}_\theta$ is strictly lower triangular in token order, despite using fresh cache entries within a round.

If $C = \widehat{G}_\theta(C)$, every fresh layer cache C_i^+ equals C_i. The recirculated equations then use exactly the earlier cache entries in the sequential definition of G in Section 3.1. The source states and mixed states also agree, so $C = G(C)$. Conversely, suppose $C = G(C)$. Then the source states and mixed inputs of the two maps coincide. By induction on affected layers, the same input gives the same fresh keys and values, which equal C_i because $C = G(C)$. Attention therefore reads the same earlier entries and produces the same next state. Thus $C = \widehat{G}_\theta(C)$. This proves equality of fixed points, not equality of the two maps away from equilibrium.

Strict causality proves Lemma 1 by induction on rounds. At round one, column zero has no cache predecessors and equals its fixed-point value. If columns $t < k$ are exact at round k, every predecessor of $t \le k$ is exact, so the next update gives exact columns $t < k+1$. The source pass of round

k reads $C^{(k)}$, hence is exact at positions $t \le k$. The unique fixed point is the sequential cache, and $K = T$ suffices for the entire cache and all logits.

Assumption B.1 (Local bounds on the evaluated computation)**.** *Fix $\bar{\varepsilon} > 0$ and a convex cache neighborhood $\mathcal{U}$. The source and recirculated passes in Eq. (2) are well-defined for every $C \in \mathcal{U}$ and $0 \le \varepsilon \le \bar{\varepsilon}$. On domains containing the intermediate states and cache arguments of these passes, the following bounds hold for all required comparison pairs. These include source pairs $(h, \mathsf{C}_i), (h', \mathsf{C}'_i)$ and recirculated graph pairs $(h, V_i(h)), (h', V_i(h'))$. If derivative bounds are used to obtain the inequalities, the domains must also contain the connecting comparison segments, including those between the plain and mixed computations:*

1. *The layer sensitivity constants a_i, b_i, κ_i introduced in Section 3.3 are nonnegative and satisfy*
$$\begin{aligned} \|F_i(h, c) - F_i(h', c')\|_{\mathrm{tok}} &\le a_i \|h - h'\|_{\mathrm{tok}} + b_i \|c - c'\|_{\mathrm{tok}}, \\ \|V_i(h) - V_i(h')\|_{\mathrm{tok}} &\le \kappa_i \|h - h'\|_{\mathrm{tok}}. \end{aligned} \tag{4}$$
2. *With z_d fixed, the mixer sensitivity constant λ satisfies*
$$\|H_\theta(z_d, u) - H_\theta(z_d, u')\|_{\mathrm{tok}} \le \lambda \|u - u'\|_{\mathrm{tok}}. \tag{5}$$
3. *The domains and constants above are chosen before running Jacobi iteration. Whenever a result is asserted uniformly over sequence lengths, the constants, admissible mixing range, and specified neighborhood radii are uniform over those lengths and inputs.*

Remark (Lipschitz bounds for neural networks). Lipschitz bounds are widely used in the theoretical analysis of deep neural networks. For example, bounds based on the spectral norms of layer weights appear in generalization guarantees (Bartlett et al., 2017), while Lipschitz bounds also provide certificates of robustness to input perturbations (Hein & Andriushchenko, 2017; Tsuzuku et al., 2018). Assumption B.1 uses local bounds of this kind to control how changes in residual states and cached keys and values affect the layer and mixer computations.

The assumption is local and permits $a_i > 1$, so individual layers need not contract. It describes a neighborhood on which the maps can be evaluated, rather than assuming that Jacobi iterates remain there. Invariance is established below. Bounded first derivatives on appropriate convex domains imply Eq. (4) by integration along a segment. We next give more explicit sufficient bounds for attention and mixing. This avoids assuming a contraction bound for the complete map.

B.1.2 Obtaining the local constants from transformer operations

The constants of Assumption B.1 can be bounded from the weights and states of the model, as this subsection shows. Its main point is that attention contributes no factor of the sequence length, which the length-independent bounds of Appendix B.1.6 require.

Lemma B.2 (Attention sensitivity without a token-count factor)**.** *Consider a single attention head at a fixed query position t, with scores $\tau\langle q, k_j\rangle$ and output $o = \sum_{j \in S_\ell(t)} \pi_j v_j$, where π is the softmax of the scores over the attended positions $S_\ell(t) \subseteq \{0, \dots, t\}$ of its layer (Section 2), which contain t, and $\tau > 0$ is the score scale. Suppose both compared computations satisfy $\|q\|_2 \le \bar{q}$, $\|k_j\|_2 \le \bar{k}$ and $\|v_j\|_2 \le \bar{v}$. Suppose also that the query, key and value of the current position, as functions of its input state h, have Lipschitz constants L_q, L_k, L_v. If the input-state difference is at most δ_h and, at every attended position $j < t$, the cached key and the cached value each differ by at most e, then*
$$\|o - o'\|_2 \le a_{\mathrm{att}} \delta_h + b_{\mathrm{att}} e, \qquad \begin{aligned} a_{\mathrm{att}} &= L_v + \bar{v}\tau(\bar{k} L_q + \bar{q} L_k), \\ b_{\mathrm{att}} &= 1 + \bar{v}\tau\bar{q}. \end{aligned} \tag{6}$$
These constants do not depend on the number of attended positions.

Proof. Let π be a probability vector. Let J be a random index with $\Pr(J = j) = \pi_j$ and set $X := x_J$, so that $\mathbb{E}X = \langle \pi, x\rangle$. Cauchy–Schwarz and $\mathbb{E}(X - \mathbb{E}X)^2 \le \mathbb{E}X^2 \le \|x\|_\infty^2$ give
$$\|(\mathrm{diag}(\pi) - \pi\pi^{\mathsf{T}})x\|_1 = \textstyle\sum_j \pi_j \big| x_j - \langle \pi, x\rangle \big| = \mathbb{E}|X - \mathbb{E}X| \le \sqrt{\mathbb{E}(X - \mathbb{E}X)^2} \le \|x\|_\infty.$$
Integration along the segment between two score vectors ψ, ψ' yields $\|\,\mathrm{softmax}(\psi) - \mathrm{softmax}(\psi')\|_1 \le \|\psi - \psi'\|_\infty$. Both compared computations attend over the same positions $S_\ell(t)$,

which do not depend on the states or the cache. The bound therefore applies to the score vectors restricted to $S_\ell(t)$, which include the current position t.

At an attended position $j < t$, the key is read from the cache and differs by at most e. At $j = t$, it is computed from the current input state and differs by at most $L_k\delta_h$. Thus the maximum key difference over $S_\ell(t)$ is at most $e + L_k\delta_h$. Expanding the difference of the two inner products, and using the bounds on queries and keys at the two endpoints, gives $\|\psi - \psi'\|_\infty \le \tau\big((\bar{k}L_q + \bar{q}L_k)\delta_h + \bar{q}e\big)$. The maximum value difference is similarly at most $e + L_v\delta_h$. Finally, $\|\sum_j \pi_j v_j - \sum_j \pi'_j v'_j\|_2 \le \max_j \|v_j - v'_j\|_2 + \bar{v}\|\pi - \pi'\|_1$. Combining the three estimates proves Eq. (6). Normalization of the attention weights, rather than a sum of worst-case errors over positions, removes any factor proportional to sequence length. □

From heads to a complete layer.

Let $\mathrm{att}(h, c)$ denote the multi-head attention output after the output projection W_O. Its arguments are the current token's input state h and the cached keys and values c of earlier tokens. The query, key and value of the current token are computed from h. Index the heads by n and let $a_{\mathrm{att},n}, b_{\mathrm{att},n}$ be the constants of Lemma B.2 for head n. We define the attention-branch constants

$$a_{\mathrm{br}} := \|W_O\|_2 \left(\sum_n a_{\mathrm{att},n}^2\right)^{1/2}, \qquad b_{\mathrm{br}} := \|W_O\|_2 \left(\sum_n b_{\mathrm{att},n}^2\right)^{1/2}.$$

Then $\|\mathrm{att}(h, c) - \mathrm{att}(h', c')\|_2 \le a_{\mathrm{br}}\delta_h + b_{\mathrm{br}}e$. By Lemma B.2 and the triangle inequality in ℓ_2 over heads, the concatenated head outputs differ by at most $(\sum_n a_{\mathrm{att},n}^2)^{1/2}\delta_h + (\sum_n b_{\mathrm{att},n}^2)^{1/2}e$, and W_O multiplies this by at most $\|W_O\|_2$. In the residual-block example below, FFN denotes the feedforward map and Norm the normalization map, with local Lipschitz constants L_{FFN} and L_{Norm}, respectively. For example, the residual block $y = h + \mathrm{att}(h, c)$, $F(h, c) = y + \mathrm{FFN}(\mathrm{Norm}(y))$ satisfies Eq. (4) with

$$a = (1 + L_M L_{\mathrm{Norm}})(1 + a_{\mathrm{br}}), \qquad b = (1 + L_M L_{\mathrm{Norm}})b_{\mathrm{br}}, \qquad \kappa = \left(\sum_n (L_{k,n}^2 + L_{v,n}^2)\right)^{1/2}. \tag{7}$$

To see this, apply the branch bound at every position with $\delta_h = \|h - h'\|_{\mathrm{tok}}$ and $e = \|c - c'\|_{\mathrm{tok}}$. The skip connection around attention, $y = h + \mathrm{att}(h, c)$, and the skip connection around the feedforward map, $F = y + \mathrm{FFN}(\mathrm{Norm}(y))$, each split a difference into the skipped input and the branch output, so

$$\|y - y'\|_2 \le \|h - h'\|_2 + \|\mathrm{att}(h, c) - \mathrm{att}(h', c')\|_2 \le (1 + a_{\mathrm{br}})\delta_h + b_{\mathrm{br}}e,$$
$$\|F - F'\|_2 \le \|y - y'\|_2 + \|\mathrm{FFN}(\mathrm{Norm}(y)) - \mathrm{FFN}(\mathrm{Norm}(y'))\|_2 \le (1 + L_M L_{\mathrm{Norm}})\|y - y'\|_2.$$

The fresh key and value of head n change by at most $L_{k,n}\delta_h$ and $L_{v,n}\delta_h$, and concatenating them over heads gives κ. The constants of the normalization before attention and of any query and key normalization are included in L_q, L_k, L_v. Equation (7) is a worked block example. Additional normalizations or residual scalings in a particular transformer are handled by composing their bounds in their actual order. In this example $a \ge 1$. If $a_{p+1} \ge 1$, adding layer $p + 1$ to the loop does not decrease $\prod_i (a_i + b_i\kappa_i) - \prod_i a_i$, so χ is nondecreasing in p for fixed ε and λ.

These constants follow from standard operator bounds. An affine map $h \mapsto Wh + b$ has Lipschitz constant $\|W\|_2$. For a feedforward map $\mathrm{FFN}(h) = W_2\phi(W_1h + b_1) + b_2$, where ϕ acts elementwise with Lipschitz constant L_ϕ, composing these bounds gives $L_{\mathrm{FFN}} \le \|W_2\|_2 L_\phi \|W_1\|_2$. For a product of two bounded branches, the inequality $\|u \odot v - u' \odot v'\|_2 \le \|u\|_\infty \|v - v'\|_2 + \|v'\|_\infty \|u - u'\|_2$ gives the corresponding gated-feedforward bound. For a normalization scale vector γ_{Norm} and a stabilizing constant $\delta > 0$, define $r(h) := \sqrt{\|h\|_2^2/H + \delta}$. If $\mathrm{Norm}(h) = \gamma_{\mathrm{Norm}} \odot h/r(h)$, differentiation of $h/r(h)$ gives an operator norm at most $1/r(h) \le 1/\sqrt{\delta}$, and hence $L_{\mathrm{Norm}} \le \|\gamma_{\mathrm{Norm}}\|_\infty/\sqrt{\delta}$. Layer normalization has the same bound after composing with the orthogonal centering projection. These bounds can be conservative, but they are expressed in model quantities and do not multiply by the number of tokens.

The norm-matching mixer. Suppose $\|z_d\|_2 \le D_d$, and let u and u' be two source states with norms at least $r_s > 0$. Adding and subtracting $u'/\|u\|_2$ shows that $\|u/\|u\|_2 - u'/\|u'\|_2\|_2 \le (2/r_s)\|u - u'\|_2$. Consequently $\|f(u)\|_2 \le D_d$ and $\|f(u) - f(u')\|_2 \le L_f\|u - u'\|_2$ with $L_f = 2D_d/r_s$. The fixed convex and additive rules therefore satisfy Eq. (5) with $\lambda = L_f$. If, for the adaptive family in Eq. (3), $\|a_\theta\|_\infty \le \bar{a}$, $\|b_\theta\|_\infty \le \bar{b}$, and their source-input Lipschitz constants into Euclidean norm

are L_a, L_b, then

$$\lambda \le D_d L_b + D_d L_a + \bar{a} L_f, \qquad \|H_\theta(z_d, z_s)\|_2 \le D_d(\bar{a} + \bar{b}). \tag{8}$$

To prove the first inequality, split the differences of $b_\theta \odot z_d$ and $a_\theta \odot f$ into gate differences and source-vector differences, and apply the product inequality above. The second follows directly from the bounds on the gate values. For the sigmoid mixer of Section 2, $\bar{a}, \bar{b} \le 1/\varepsilon$, and L_a, L_b include the weight matrices of its MLP.

The lower bound r_s is needed because the norm matching f divides by $\|z_s\|_2$. It must hold throughout the compared source states. The stabilizing constant δ of the normalization does not provide it. For fixed T, a positive minimum source norm at the plain pass extends to a sufficiently small neighborhood by continuity. For the sequence-length-independent bounds in Corollary B.7, this argument requires a common lower bound $r_s > 0$ on the source-state norms throughout the relevant neighborhoods, independent of the sequence lengths and inputs considered.

B.1.3 Separating upward propagation from feedback

To bound one complete cache update, we distinguish two computations along which perturbations propagate. Within the recirculated pass, changes in freshly written keys and values affect higher layers. Through the feedback path, changes in the supplied cache affect the source state, which the mixer then feeds back into the destination layer. Lemma B.3 bounds these two effects separately and combines them into a layerwise inequality. This will allow us to derive a contraction condition for the complete update even when individual layers amplify perturbations.

Lemma B.3 (Explicit layer gain matrix)**.** *Under Assumption B.1, use the convention that an empty product is one and define*

$$\begin{aligned} v_r &= \begin{cases} b_r \prod_{j=r+1}^{p} a_j, & r \le p, \\ 0, & r > p, \end{cases} \qquad g_i = \kappa_i \prod_{j=1}^{i-1} a_j, \\ A_{ir} &= \begin{cases} \kappa_i b_r \prod_{j=r+1}^{i-1} a_j, & r < i, \\ 0, & r \ge i, \end{cases} \quad B_{ir} = \lambda g_i v_r. \end{aligned} \tag{9}$$

Then A, B are nonnegative and A is strictly lower triangular. For $C, C' \in \mathcal{U}$, define $e_r = \|\mathsf{C}_r - \mathsf{C}'_r\|_{\mathrm{tok}}$ and $e_i^+ = \|[\widehat{G}_\theta(C)]_i - [\widehat{G}_\theta(C')]_i\|_{\mathrm{tok}}$. Componentwise,

$$e^+ \le Ae^+ + \varepsilon Be, \qquad e^+ \le \varepsilon (I - A)^{-1} Be. \tag{10}$$

Proof. Put $\Delta S_i = \|S_i - S'_i\|_{\mathrm{tok}}$. The source passes have the same initial state, so $\Delta S_0 = 0$, and Eq. (4) gives $\Delta S_i \le a_i \Delta S_{i-1} + b_i e_i$. Expanding this recursion up to the source layer yields

$$\Delta S_p \le \textstyle\sum_{r=1}^{p} b_r \left(\prod_{j=r+1}^{p} a_j \right) e_r = \sum_{r=1}^{m} v_r e_r. \tag{11}$$

The mixed starting states thus satisfy $\Delta U_0 \le \varepsilon \lambda \sum_r v_r e_r$. For the recirculated pass, the same layer bound yields $\Delta U_i \le a_i \Delta U_{i-1} + b_i e_i^+$, because the supplied cache in this pass is the freshly produced output. Expanding up to the input of layer i gives

$$\Delta U_{i-1} \le \left(\textstyle\prod_{j=1}^{i-1} a_j \right) \Delta U_0 + \sum_{r<i} b_r \left(\prod_{j=r+1}^{i-1} a_j \right) e_r^+ .$$

The new cache at layer i is extracted from this input, so its difference is at most $\kappa_i \Delta U_{i-1}$. Substitution gives $e^+ \le Ae^+ + \varepsilon Be$. Since $A^m = 0$, repeated substitution eliminates $A^m e^+$ and yields $e^+ \le \varepsilon \sum_{j=0}^{m-1} A^j Be = \varepsilon (I - A)^{-1} Be$. The terms in A propagate output-cache differences through earlier *layers* of the current recirculated pass. The terms in εB pass through the source state and the mixer. No assumption that $a_i < 1$ was used. □

All upward-path constants are taken uniformly over the source and recirculated states in the chosen neighborhood.

B.1.4 The local contraction theorem

Lemma B.3 bounds cache differences separately at each affected layer. We now combine these bounds using a weighted norm that accounts for amplification as perturbations propagate up the

stack. Theorem B.4 shows that, when mixing is sufficiently weak, the complete cache update contracts in this norm. Consequently, if a fixed point lies in the interior of the assumed neighborhood, iterations initialized sufficiently close to it converge geometrically.

Theorem B.4 (Contraction in a norm weighted by layer)**.** *Under Assumption B.1, let A, B be given by Eq. (9) and define*

$$\begin{aligned} w &= (I-A)^{-1}\mathbf{1} = \textstyle\sum_{j=0}^{m-1} A^j\mathbf{1}, \qquad \eta = \max_i (Bw)_i, \\ \|C\|_w &= \max_i \tfrac{\|\mathsf{C}_i\|_{\mathrm{tok}}}{w_i}, \qquad\qquad w_{\max} = \max_i w_i. \end{aligned} \tag{12}$$

If $0 \le \varepsilon \le \bar{\varepsilon}$ and $\varepsilon\eta < 1$, then

$$\|\widehat{G}_\theta(C) - \widehat{G}_\theta(C')\|_w \le \widehat{q}_\varepsilon \|C - C'\|_w, \qquad \widehat{q}_\varepsilon = \varepsilon\eta < 1, \tag{13}$$

for all $C, C' \in \mathcal{U}$. In particular, if a fixed point $C^\star_\varepsilon$ belongs to the interior of $\mathcal{U}$, every sufficiently small closed weighted ball centered there is invariant, and any initialization in that ball satisfies

$$\|C^{(k)} - C^\star_\varepsilon\|_w \le \widehat{q}_\varepsilon^{\,k} \|C^{(0)} - C^\star_\varepsilon\|_w. \tag{14}$$

Proof. Strict lower triangularity gives $A^m = 0$, so $(I-A)^{-1} = \sum_{j=0}^{m-1} A^j$ is nonnegative and $w_i \ge 1$. For any two caches, $e \le w\|C - C'\|_w$. Moreover, $Bw \le \eta\mathbf{1}$, hence $(I-A)^{-1}Bw \le \eta(I-A)^{-1}\mathbf{1} = \eta w$. Lemma B.3 now gives $e^+ \le \varepsilon(I-A)^{-1}Bw\|C - C'\|_w \le \varepsilon\eta w\|C - C'\|_w$. Divide each component by w_i and take the maximum to obtain Eq. (13). Setting $C' = C^\star_\varepsilon$ proves invariance of every sufficiently small weighted ball about the fixed point contained in $\mathcal{U}$. Iteration proves Eq. (14). □

Relation to the original cache norm. Let $w_{\min} = \min_i w_i$. Equation (1) gives $w_{\min}\|C\|_w \le \|C\|_{\mathrm{blk}} \le w_{\max}\|C\|_w$. If the per-token norm in Section 3.3 is Euclidean on the concatenated layer blocks, then $\|C\|_{\mathrm{blk}} \le \|C\|_\infty \le \sqrt{m}\,\|C\|_{\mathrm{blk}}$. Both inequalities follow by comparing the largest block norm with the Euclidean norm of their concatenation, first at each token and then taking a maximum over tokens. Hence

$$\|C^{(k)} - C^\star_\varepsilon\|_\infty \le \tfrac{\sqrt{m}\, w_{\max}}{w_{\min}} \widehat{q}_\varepsilon^{\,k} \|C^{(0)} - C^\star_\varepsilon\|_\infty. \tag{15}$$

The theorem therefore establishes geometric convergence in the original norm with a prefactor. It does not assert one-step contraction in that unweighted norm.

The condition of Proposition 1. Take the column norm $\|C_t\|_w := \max_i \|\mathsf{C}_{i,t}\|_2/w_i$, so that $\|C\|_w = \max_t \|C_t\|_w$. Given C, C' and t, let C'' agree with C' at positions $j < t$ and with C at $j \ge t$. Since $\widehat{G}_{\theta,t}$ depends only on $C_{<t}$, Eq. (13) applied to C and C'' gives $\|\widehat{G}_{\theta,t}(C_{<t}) - \widehat{G}_{\theta,t}(C'_{<t})\|_w \le \widehat{q}_\varepsilon \max_{j<t}\|C_j - C'_j\|_w$ whenever $C'' \in \mathcal{U}$. Hence, on any $\|\cdot\|_w$-ball about $C^\star_\varepsilon$ contained in $\mathcal{U}$, Proposition 1 holds with $\rho = \widehat{q}_\varepsilon$.

B.1.5 A BASIN CONTAINING THE PLAIN CACHE

Theorem B.4 guarantees convergence when the initial cache is sufficiently close to a fixed point. However, the algorithm starts from C^{pl}, and we do not yet know whether this cache is close enough. We now give conditions that guarantee convergence from this starting point. When $\varepsilon = 0$, mixing is switched off. The recirculated pass then starts from z_d and recomputes its keys and values exactly as in a plain forward pass, so $\widehat{G}_\theta(C) = C^{\mathrm{pl}}$ for every admissible C.

Lemma B.5 first bounds how far one cache update moves the plain cache. Theorem B.6 then combines this bound with contraction to give an explicit condition under which a neighborhood of C^{pl} contains a unique fixed point and all subsequent iterates. It also bounds the distance from the plain cache to that fixed point and the geometric decay of the iteration error.

Lemma B.5 (Defect of the plain cache)**.** *Let $S^{\mathrm{pl}}_p := S_p(C^{\mathrm{pl}})$ denote the source-state array from the plain forward pass. We compare this plain pass, starting from z_d, with the recirculated pass starting from $z_d + \varepsilon H_\theta(z_d, S^{\mathrm{pl}}_p)$. Suppose the layer sensitivity bounds in Eq. (4) hold for the pairs of layer inputs and cache arguments arising in this comparison. Choose an upper bound h_0 and define $\widehat{g}_i, \widehat{D}$ by*

$$\begin{aligned} &h_0 \ge \|H_\theta(z_d, S^{\mathrm{pl}}_p)\|_{\mathrm{tok}}, \\ &\widehat{g}_i = \kappa_i \textstyle\prod_{j=1}^{i-1}(a_j + b_j\kappa_j), \qquad \widehat{D} = h_0 \max_i \tfrac{\widehat{g}_i}{w_i}. \end{aligned} \tag{16}$$

Here $a_j + b_j\kappa_j$ bounds propagation through a layer whose attention reads the keys and values computed in the same pass. Then

$$\|\widehat{G}_\theta(C^{\mathrm{pl}}) - C^{\mathrm{pl}}\|_w \le \varepsilon\widehat{D}. \tag{17}$$

Proof. Compare the mixed recirculated pass to the plain pass. Each pass reads the keys and values it computes itself, so their cache arguments differ. Their initial states differ by at most εh_0, since the source input is $S_p(C^{\mathrm{pl}}) = S_p^{\mathrm{pl}}$. If their input-state difference at layer i is d_{i-1}, then the KV difference is at most $\kappa_i d_{i-1}$ and $d_i \le a_i d_{i-1} + b_i\kappa_i d_{i-1} = (a_i + b_i\kappa_i)d_{i-1}$. Induction gives $d_{i-1} \le \varepsilon h_0 \prod_{j<i}(a_j + b_j\kappa_j)$ and cache difference at most $\varepsilon h_0 \widehat{g}_i$. Dividing by w_i and taking the maximum proves the claim. □

Theorem B.6 (Convergence from plain-cache initialization)**.** *Let $R > 0$, and suppose Assumption B.1 holds on an open convex neighborhood of the closed ball $\mathcal{B}_R = \{C : \|C - C^{\mathrm{pl}}\|_w \le R\}$, with constants valid for $0 \le \varepsilon \le \bar{\varepsilon}$. Let $\widehat{D}$ be as in Lemma B.5, and fix ε in this range. If*

$$\varepsilon\eta < 1, \qquad \varepsilon\widehat{D} \le (1 - \widehat{q}_\varepsilon)R, \tag{18}$$

then $\mathcal{B}_R$ is invariant under $\widehat{G}_\theta$ and contains a unique fixed point $C^\star_\varepsilon$. This is the cache of sequential recirculation for the same input and mixing rule. Moreover,

$$\|C^\star_\varepsilon - C^{\mathrm{pl}}\|_w \le \tfrac{\varepsilon\widehat{D}}{1-\widehat{q}_\varepsilon}, \qquad \|C^{(k)} - C^\star_\varepsilon\|_w \le \widehat{q}_\varepsilon^{\,k}\, \tfrac{\varepsilon\widehat{D}}{1-\widehat{q}_\varepsilon} \quad (C^{(0)} = C^{\mathrm{pl}}). \tag{19}$$

Both inequalities in Eq. (18) hold whenever

$$\varepsilon\big(\widehat{D} + \eta R\big) < R, \qquad 0 \le \varepsilon \le \bar{\varepsilon}. \tag{20}$$

Proof. For $C \in \mathcal{B}_R$, Theorem B.4 and Lemma B.5 give

$$\|\widehat{G}_\theta(C) - C^{\mathrm{pl}}\|_w \le \|\widehat{G}_\theta(C) - \widehat{G}_\theta(C^{\mathrm{pl}})\|_w + \|\widehat{G}_\theta(C^{\mathrm{pl}}) - C^{\mathrm{pl}}\|_w \le \widehat{q}_\varepsilon R + \varepsilon\widehat{D} \le R.$$

This proves invariance rather than assuming boundedness of the iterates. The ball is closed in a finite-dimensional normed space and hence complete. By the Banach fixed-point theorem, it contains a unique fixed point of $\widehat{G}_\theta$, and the iterates from any point in the ball converge to it.

The fixed-point equivalence proved in Appendix B.1.1 identifies this fixed point with the sequential cache. Strict token causality makes that solution unique wherever the token computations are defined. Applying the preceding inequality to the fixed point itself gives $\|C^\star_\varepsilon - C^{\mathrm{pl}}\|_w \le \widehat{q}_\varepsilon\|C^\star_\varepsilon - C^{\mathrm{pl}}\|_w + \varepsilon\widehat{D}$. Rearranging proves the first bound in Eq. (19), and contraction proves the second. Finally, Eq. (20) implies $\varepsilon\eta < 1$ and $\varepsilon\widehat{D} < (1 - \varepsilon\eta)R$, which proves Eq. (18). □

Choosing a neighborhood of the plain cache. Theorem B.6 requires the layer and mixer bounds to hold on a ball around C^{pl}. Start with the plain forward pass and bound each layer's sensitivity when its input states and cache entries are close to their plain-pass values. Use these bounds to construct A, B, w. Then choose the radius R small enough that, for any cache in the ball, one update keeps the source and recirculated states within the regions where these bounds hold.

For fixed T, continuity allows this choice when ε is sufficiently small, provided the plain-pass states stay away from points where the computations are undefined. This checks what happens during one update. If the remaining conditions of Theorem B.6 also hold, the theorem guarantees that every later iterate stays in the ball.

For example, we can check that the source norms remain positive. By Eq. (11), every $C \in \mathcal{B}_R$ satisfies $\|S_p(C) - S_p(C^{\mathrm{pl}})\|_{\mathrm{tok}} \le R\sum_r v_r w_r$. Suppose the smallest source norm in the plain pass is $r_0 := \min_t \|S^{\mathrm{pl}}_{p,t}\|_2 > 0$. If we choose R so that $R\sum_r v_r w_r \le r_0/2$, every source norm in the ball is at least $r_0/2$. We can therefore take $r_s = r_0/2$.

The source-pass bounds can be obtained before bounding the mixer, because the source pass does not use the mixer. Other intermediate states can be checked in the same way.

B.1.6 SEQUENCE LENGTH, READOUT, AND THE LIMITS OF THE RESULT

We now explain what the bounds imply for longer sequences and for the algorithm we implement. Corollary B.7 states when the cache bounds are independent of sequence length and shows how cache error bounds the error in the returned logits. Figure 2(D) finds that the prediction error after two rounds does not grow with the sequence length, and the corollary gives conditions under which the theory predicts this.

Corollary B.7 (Uniform constants and source-pass readout)**.** *Consider inputs of different lengths with fixed layer geometry. Suppose the hypotheses of Theorem B.6 hold with common bounds $a_i, b_i, \kappa_i, \lambda, h_0, R, \bar{\varepsilon}$, and choose a fixed ε satisfying Eq. (20). Then the weights, contraction factor and error prefactors in Eqs. (15) and (19) are independent of sequence length.*

If the final normalization and LM head together have Lipschitz constant L_{head} on the source-state neighborhood, define

$$L_{R,w} = L_{\mathrm{head}} \sum_{r=1}^{m} b_r \left(\prod_{j=r+1}^{m} a_j \right) w_r. \tag{21}$$

The source logits $\ell^{(k)} = R(C^{(k)})$ of Jacobi round $k \geq 1$ obey

$$\max_t \|\ell_t^{(k)} - \ell_t^\star\|_2 \leq L_{R,w} \widehat{q}_\varepsilon^{\,k} \tfrac{\varepsilon \widehat{D}}{1-\widehat{q}_\varepsilon}. \tag{22}$$

Proof. Every entry of A, B in Eq. (9) is a finite sum or product of the common layer constants. The definitions of $w, \eta, \widehat{D}$ and $\widehat{q}_\varepsilon$ therefore have no dependence on T, and the norm-comparison factor in Eq. (15) depends only on these weights and m. For the readout, repeat the source-pass expansion in Eq. (11) up to m instead of p. Since $e_r \leq w_r \|C - C'\|_w$, this gives the Lipschitz constant in Eq. (21) after the final head. The source pass of round k reads $C^{(k)}$, so $\ell^{(k)} = R(C^{(k)})$ and its error is at most $L_{R,w} \widehat{q}_\varepsilon^{\,k} \|C^{(0)} - C_\varepsilon^\star\|_w$. Substituting the plain-initialization bound proves Eq. (22). □

Proof of the contraction guarantee. To prove Proposition 1, note that $C^\star = \widehat{G}_\theta(C^\star)$ by the fixed-point equivalence of Appendix B.1.1. Maximizing its hypothesis over t gives $\|\widehat{G}_\theta(C) - \widehat{G}_\theta(C')\|_\infty \leq \rho \|C - C'\|_\infty$ on the ball. Taking $C' = C^\star$ gives invariance and, by induction, $\|C^{(k)} - C^\star\|_\infty \leq \rho^k \|C^{(0)} - C^\star\|_\infty$. Since $\ell^{(k)} = R(C^{(k)})$ and $\ell^\star = R(C^\star)$, the Lipschitz bound on R_t gives the logit bound.

The uniformity hypothesis includes the radii and normalization margins, not just the displayed formulas. Attention permits constants without a factor T by Lemma B.2. A uniform bound on the states visited across the chosen family of inputs must still be checked. Products of layer constants can make the sufficient threshold conservative, and the theorem does not assert that two rounds suffice.

Finally, the additional layer information matters. For fixed T, token triangularity alone makes the Jacobian nilpotent, but does not provide a useful uniform maximum-norm contraction. For example, the triangular map $\Phi(c_0, \ldots, c_{T-1}) = (0, c_0, \ldots, c_{T-2})$ has the unique fixed point zero, while iteration from the all-ones vector has maximum-norm error one for every $k < T$. A contracting token-weighted norm can hide this delay in weights growing geometrically with T. The weights in Eq. (12) instead depend on layer gains; under the explicit common bounds of Corollary B.7, their norm-equivalence constants remain controlled as the sequence grows.

B.2 ACCURACY OF THE UNROLLED TRAINING GRADIENT

During training, the cache obtained after K Jacobi rounds is used to compute the prediction loss. Appendix B.1 bounds the difference between this cache and the cache produced by sequential recirculation. We now ask whether differentiating these K rounds also gives a loss gradient close to that of sequential recirculation.

The cache-error bound is one part of this analysis. We must also control how the cache changes with the mixer parameters. Two caches can be close while their parameter derivatives differ substantially. We therefore combine the bounds from Appendix B.1 with additional derivative bounds to control the error in the training gradient. This proves the bound of Section 3.3. Figure 4(C, D) measures the

same gradient error during mixer training. We consider exact arithmetic, a fixed number of rounds, and differentiation through every cache update $C^{(k)} = \widehat{G}_\theta(C^{(k-1)})$.

B.2.1 Objective, norms, and additional local assumptions

Fix an input sequence and its target tokens, the frozen transformer, and the mixing strength ε of the family in Eq. (3). Only the mixer parameters $\theta \in \mathbb{R}^{n_\theta}$ vary. We drop the subscript ε used in Appendix B.1 and define

$$C^{(0)} = C^{\mathrm{pl}}, \qquad C^{(k+1)}(\theta) = \widehat{G}_\theta(C^{(k)}(\theta)), \qquad C^\star(\theta) = \widehat{G}_\theta(C^\star(\theta)). \tag{23}$$

The plain-cache initialization is independent of θ, so $D_\theta C^{(0)} = 0$.

We use one fixed layer-weighted cache norm $\|\cdot\|_w$ from Eq. (12), and the Euclidean norm on parameter perturbations. For linear maps on caches and from parameters to caches, respectively, we use $\|J\|_{w\leftarrow w} = \sup_{\|u\|_w \le 1} \|Ju\|_w$ and $\|P\|_{w\leftarrow 2} = \sup_{\|v\|_2 \le 1} \|Pv\|_w$. For a differentiable scalar function a of the cache, we measure its derivative at C using the dual norm associated with $\|\cdot\|_w$ (Boyd & Vandenberghe, 2004, Section A.1.6): $\|Da(C)\|_{w,*} := \sup_{\|u\|_w \le 1} |Da(C)[u]|$. Here $Da(C)[u]$ is the directional derivative of a at C in the direction u. Below, a norm of a cache Jacobian or a parameter-to-cache sensitivity means the corresponding induced norm above.

Assumption B.8 requires the updates to stay in the same cache neighborhood and satisfy a common contraction bound as the parameters vary. It also bounds the derivatives of the update and how much they change when the cache changes.

Assumption B.8 (A common contractive neighborhood and derivative regularity)**.** *Let Θ be an open convex neighborhood of the parameter value under consideration, and let $\mathcal{B}_R = \{C : \|C - C^{\mathrm{pl}}\|_w \le R\}$ be a fixed closed cache ball of positive radius. The following hold.*

1. *The map $(\theta, C) \mapsto \widehat{G}_\theta(C)$ is jointly continuously differentiable on an open set containing $\Theta \times \mathcal{B}_R$. For every $\theta \in \Theta$, $\widehat{G}_\theta(\mathcal{B}_R) \subseteq \mathcal{B}_R$ and, for a common $0 < q < 1$,*

$$\|D_C \widehat{G}_\theta(C)\|_{w\leftarrow w} \le q \quad \text{for all } C \in \mathcal{B}_R. \tag{24}$$

2. *There are finite constants $M_\theta, L_{CC}, L_{C\theta}$ such that, for all $\theta \in \Theta$ and $C, C' \in \mathcal{B}_R$,*

$$\begin{aligned} \|D_\theta \widehat{G}_\theta(C)\|_{w\leftarrow 2} &\le M_\theta, \\ \|D_C \widehat{G}_\theta(C) - D_C \widehat{G}_\theta(C')\|_{w\leftarrow w} &\le L_{CC}\|C - C'\|_w, \\ \|D_\theta \widehat{G}_\theta(C) - D_\theta \widehat{G}_\theta(C')\|_{w\leftarrow 2} &\le L_{C\theta}\|C - C'\|_w. \end{aligned} \tag{25}$$

Remark (Regularity for differentiating the iteration).

To verify the contraction and invariance required in item 1 of Assumption B.8, we can apply Theorems B.4 and B.6. Take common upper bounds for the layer and mixer constants throughout Θ, construct $A, B, w, \eta, \widehat{D}$ from those bounds, and require the basin condition uniformly. Item 1 then holds with any $q \in (0, 1)$ such that $q \ge \widehat{q}_\varepsilon$. The weights w and the bound q do not depend on θ and are not differentiated.

Item 2 of Assumption B.8 is not needed for the cache convergence of Appendix B.1. Bounded second cache derivatives and mixed cache–parameter derivatives on the stated domains imply its two Lipschitz inequalities by integration along cache segments. For a fixed finite sequence, smooth attention and activations, bounded local states and parameters, stabilized normalization, and a positive lower bound on the source norm in the norm matching of the mixing rule (Section 2) provide such bounds on compact local domains.

The source pass does not apply the mixer, so the source readout $R(C) = (R_t(C_{<t}))_t$ depends on θ only through C. We call a position t *scored* if its source logits $R_t(C_{<t})$, which predict the next token x_{t+1}, contribute to the loss, and collect the scored positions in a nonempty set $\mathcal{I} \subseteq \{0, \dots, T-2\}$ with targets $y_t = x_{t+1}$. For the arXiv mixers every prediction in the window is scored, so $\mathcal{I} = \{0, \dots, T-2\}$. For the GSM8K mixers only the positions whose next token belongs to the reference solution are scored (Appendix C.6). We take the mean cross-entropy over the scored

positions,

$$h(C) = \frac{1}{|\mathcal{I}|}\sum_{t\in\mathcal{I}}\left[\log\sum_{v=1}^{V_{\mathrm{voc}}}\exp\big(R_t(C_{<t})_v\big) - R_t(C_{<t})_{y_t}\right], \qquad \mathcal{F}_k(\theta) = h(C^{(k)}(\theta)), \qquad \mathcal{F}_\star(\theta) = h(C^\star(\theta)). \tag{26}$$

Here V_{voc} is the vocabulary size. We also need bounds on the loss derivative and on how it changes with the cache.

Assumption B.9 (Loss regularity)**.** *The loss h is continuously differentiable on an open neighborhood of $\mathcal{B}_R$. There are finite constants $M_h, L_h \geq 0$ such that, for all $C, C' \in \mathcal{B}_R$,*

$$\|Dh(C)\|_{w,*} \leq M_h, \qquad \|Dh(C) - Dh(C')\|_{w,*} \leq L_h\|C - C'\|_w. \tag{27}$$

Remark (Checking the loss regularity). For the mean cross-entropy used here, Lemma B.13 gives sufficient conditions on the source readout R and derives explicit constants M_h and L_h. These conditions concern the complete loss $h(C)$, including its dependence on the neural-network readout.

B.2.2 Differentiating the fixed point and the actual unrolling

Lemma B.10 shows that the sequential cache is differentiable with respect to the mixer parameters and gives a formula for its derivative. Theorem B.11 then bounds the difference between this derivative and the one obtained by differentiating the Jacobi rounds. This supplies the derivative estimate needed for the loss-gradient analysis in Appendix B.2.3.

Lemma B.10 (Fixed-point differentiability)**.** *Under Assumption B.8, each $\theta \in \Theta$ has a unique fixed point in $\mathcal{B}_R$. It is continuously differentiable in θ, and, with $J_\star = D_C\widehat{G}_\theta(C^\star)$ and $B_\star = D_\theta\widehat{G}_\theta(C^\star)$,*

$$P_\star := D_\theta C^\star = (I - J_\star)^{-1}B_\star, \qquad \|P_\star\|_{w\leftarrow 2} \leq \overline{P} := \tfrac{M_\theta}{1-q}. \tag{28}$$

It is also the derivative computed by backpropagation through sequential recirculation.

Proof. Integrating Eq. (24) along a cache segment proves that $\widehat{G}_\theta$ is q-Lipschitz on the convex ball $\mathcal{B}_R$. Invariance and completeness give a unique fixed point there, as in Theorem B.6; the fixed-point equivalence of Appendix B.1.1 identifies it with the sequential cache. For $\theta, \theta' \in \Theta$, the two fixed-point equations give

$$\begin{aligned}\|C^\star(\theta') - C^\star(\theta)\|_w &\leq q\|C^\star(\theta') - C^\star(\theta)\|_w \\ &\quad + \|\widehat{G}_{\theta'}(C^\star(\theta)) - \widehat{G}_\theta(C^\star(\theta))\|_w \\ &\leq q\|C^\star(\theta') - C^\star(\theta)\|_w + M_\theta\|\theta' - \theta\|_2.\end{aligned}$$

The last step integrates $D_\theta\widehat{G}_\theta$ along the parameter segment in Θ. Thus $C^\star$ is Lipschitz in θ with constant $\overline{P}$.

Fix θ, let u be a small parameter increment, and write $\Delta C = C^\star(\theta + u) - C^\star(\theta)$. Joint differentiability on the open domain yields

$$\Delta C = J_\star\Delta C + B_\star u + r(u), \qquad \|r(u)\|_w = o(\|\Delta C\|_w + \|u\|_2) = o(\|u\|_2).$$

Since $\|J_\star\| \leq q < 1$, the geometric operator series gives $(I - J_\star)^{-1} = \sum_{j=0}^{\infty} J_\star^j$ with norm at most $(1-q)^{-1}$. Multiplying the preceding expansion by this inverse proves differentiability and Eq. (28), even if the fixed point lies on the boundary of $\mathcal{B}_R$. Continuity of $C^\star$, the two Jacobians, and the inverse proves continuity of $P_\star$. Finally, sequential recirculation computes $C^\star(\theta)$ for every $\theta \in \Theta$, so backpropagation through it yields $P_\star$. □

Theorem B.11 (Cache-sensitivity error of finite unrolling)**.** *Under Assumption B.8, fix $\theta \in \Theta$ and set $E_0 = \|C^{(0)} - C^\star\|_w$ and $P_k = D_\theta C^{(k)}$. For every $k \geq 1$,*

$$\|C^{(k)} - C^\star\|_w \leq E_0 q^k, \qquad \|P_k - P_\star\|_{w\leftarrow 2} \leq \overline{P}q^k + (L_{CC}\overline{P} + L_{C\theta})E_0\, kq^{k-1}. \tag{29}$$

In addition, $\|P_k\| \leq M_\theta(1 - q^k)/(1 - q) \leq \overline{P}$.

Proof. Invariance keeps every iterate in $\mathcal{B}_R$, and contraction proves the first bound by induction. Each finite iterate is continuously differentiable by the chain rule. At fixed θ, put $J_j = D_C \widehat{G}_\theta(C^{(j)})$ and $B_j = D_\theta \widehat{G}_\theta(C^{(j)})$. Differentiating the actual update gives

$$P_0 = 0, \qquad P_k = J_{k-1} P_{k-1} + B_{k-1}. \tag{30}$$

The bound on $\|P_k\|$ follows by summing $\|P_k\| \le q\|P_{k-1}\| + M_\theta$ from $P_0 = 0$. Subtract $P_\star = J_\star P_\star + B_\star$ from Eq. (30) to obtain the exact identity

$$P_k - P_\star = J_{k-1}(P_{k-1} - P_\star) + (J_{k-1} - J_\star)P_\star + (B_{k-1} - B_\star). \tag{31}$$

Write $\Delta_k = \|P_k - P_\star\|_{w\leftarrow 2}$ and $c = (L_{CC}\overline{P} + L_{C\theta})E_0$, and bound the three terms on the right of Eq. (31) in turn. Since $C^{(k-1)} \in \mathcal{B}_R$, the contraction bound (24) gives $\|J_{k-1}(P_{k-1} - P_\star)\|_{w\leftarrow 2} \le q\,\Delta_{k-1}$. The second and third inequalities of Eq. (25), together with $\|P_\star\|_{w\leftarrow 2} \le \overline{P}$ from Lemma B.10, give

$$\|(J_{k-1} - J_\star)P_\star\|_{w\leftarrow 2} + \|B_{k-1} - B_\star\|_{w\leftarrow 2} \le \left(L_{CC}\overline{P} + L_{C\theta}\right)\|C^{(k-1)} - C^\star\|_w.$$

By the first inequality in Eq. (29), proved above, $\|C^{(k-1)} - C^\star\|_w \le E_0 q^{k-1}$; for $k = 1$ this is the definition of E_0. Combining the three estimates, and using $P_0 = 0$, gives $\Delta_k \le q\Delta_{k-1} + c\,q^{k-1}$ with $\Delta_0 = \|P_\star\|_{w\leftarrow 2} \le \overline{P}$.

Iterating this scalar inequality gives $\Delta_k \le q^k\Delta_0 + c\sum_{j=1}^{k} q^{k-j}q^{j-1} \le \overline{P}q^k + c\,kq^{k-1}$, as required. □

Why the varying Jacobians matter. Unrolling Eq. (30) from $P_0 = 0$ gives

$$P_k = \textstyle\sum_{j=0}^{k-1}(J_{k-1}J_{k-2}\cdots J_{j+1})B_j, \tag{32}$$

where the product is the identity when $j = k-1$. The j-th term is the direct effect of the parameters on the update that produces $C^{(j+1)}$, namely B_j, carried forward to the final cache by the later updates $J_{j+1}, \ldots, J_{k-1}$. Backpropagating through k Jacobi rounds computes exactly this sum, with every Jacobian evaluated at the cache $C^{(j)}$ that the forward iteration actually visited. Compare it with the exact sensitivity $P_\star = (I - J_\star)^{-1}B_\star$ of Lemma B.10. Expanding the inverse as the Neumann series of its proof gives $P_\star = \sum_{j=0}^{\infty} J_\star^j B_\star$.

The j-th term is the effect of a parameter perturbation after it has passed through j further cache updates, all linearized at the fixed point $C^\star$. Because $J_\star$ is strictly lower triangular in token order, its T-th power vanishes and the series has at most T nonzero terms. Keeping only its first k terms gives the truncated series

$$\widetilde{P}_k := \textstyle\sum_{j=0}^{k-1} J_\star^j B_\star, \qquad P_\star - \widetilde{P}_k = \sum_{j=k}^{\infty} J_\star^j B_\star = J_\star^k P_\star. \tag{33}$$

Equivalently, $\widetilde{P}_k$ is the result of k steps of $\widetilde{P}_{j+1} = J_\star \widetilde{P}_j + B_\star$ from $\widetilde{P}_0 = 0$, that is, of differentiating k rounds *as if every round were linearized at the fixed point*. This is the approximation made by truncated-Neumann hypergradient methods, which evaluate all Jacobians at a single approximate fixed point (Grazzi et al., 2020; Lorraine et al., 2020).

The unrolled sensitivity (32) has the same form as $\widetilde{P}_k$ once its sum is reindexed, $\widetilde{P}_k = \sum_{j=0}^{k-1} J_\star^{k-1-j} B_\star$, but its Jacobians $J_{j+1}, \ldots, J_{k-1}$ and B_j are evaluated along the iterates rather than at $C^\star$. The two coincide when $J_j = J_\star$ and $B_j = B_\star$ for every $j < k$, for example if the iteration were started at $C^\star$. Writing $P_k - P_\star = (P_k - \widetilde{P}_k) - (P_\star - \widetilde{P}_k)$ separates the two terms of Eq. (29).

Truncation error. By Eq. (33), $\|P_\star - \widetilde{P}_k\|_{w\leftarrow 2} \le q^k\overline{P}$, the first term. It is the initial sensitivity mismatch $P_0 - P_\star = -P_\star$ carried through k linearized updates. If a differentiable parameter-dependent initializer is used, the same proof replaces $\overline{P}q^k$ by $q^k\|P_0 - P_\star\|$.

Jacobian-variation error. The difference $P_k - \widetilde{P}_k$ satisfies $P_k - \widetilde{P}_k = J_{k-1}(P_{k-1} - \widetilde{P}_{k-1}) + (J_{k-1} - J_\star)\widetilde{P}_{k-1} + (B_{k-1} - B_\star)$ with zero initial value. Since $\|\widetilde{P}_{k-1}\|_{w\leftarrow 2} \le \overline{P}$, the argument in the proof of Theorem B.11 gives $\|P_k - \widetilde{P}_k\|_{w\leftarrow 2} \le (L_{CC}\overline{P} + L_{C\theta})E_0\,kq^{k-1}$, the second term. It vanishes when the Jacobians do not vary along the iterates. Adding the two estimates recovers Eq. (29).

B.2.3 From cache sensitivities to the training loss gradient

Theorem B.12 bounds the difference between $\nabla_\theta \mathcal{F}_k$ and $\nabla_\theta \mathcal{F}_\star$. Lemma B.13 gives loss constants for the mean cross-entropy that do not grow with the number of scored positions.

Theorem B.12 (Finite-round approximation of the exact training gradient)**.** *Under Assumptions B.8 and B.9, the objectives in Eq. (26) satisfy, for all $k \geq 1$,*

$$\|\nabla_\theta \mathcal{F}_k - \nabla_\theta \mathcal{F}_\star\|_2 \leq A_{\mathrm{g}} q^k + B_{\mathrm{g}} k q^{k-1}, \tag{34}$$

where

$$A_{\mathrm{g}} = \overline{P}(M_h + L_h E_0), \qquad B_{\mathrm{g}} = M_h(L_{CC}\overline{P} + L_{C\theta})E_0. \tag{35}$$

Proof. The chain rule gives $D_\theta \mathcal{F}_k = Dh(C^{(k)})P_k$ and $D_\theta \mathcal{F}_\star = Dh(C^\star)P_\star$. Their difference is

$$\begin{aligned} D_\theta \mathcal{F}_k - D_\theta \mathcal{F}_\star = {} & Dh(C^{(k)})(P_k - P_\star) \\ & + \big(Dh(C^{(k)}) - Dh(C^\star)\big)P_\star. \end{aligned} \tag{36}$$

For $C \in \mathcal{B}_R$, a parameter-to-cache map P and $v \in \mathbb{R}^{n_\theta}$, we have $|Dh(C)[Pv]| \leq \|Dh(C)\|_{w,*}\|P\|_{w\leftarrow 2}\|v\|_2$. Taking the supremum over $\|v\|_2 \leq 1$ in Eq. (36) gives

$$\|\nabla_\theta \mathcal{F}_k - \nabla_\theta \mathcal{F}_\star\|_2 \leq M_h\|P_k - P_\star\| + L_h E_0 q^k \overline{P}.$$

Substitution of Eq. (29) proves the theorem. □

In A_{g}, the part $\overline{P}M_h$ is the truncation error of Eq. (33). It arises because the plain cache has zero parameter sensitivity, and it remains when $E_0 = 0$. The part $\overline{P}L_h E_0$ comes from the forward error of the cache, and B_{g} from evaluating the Jacobians at inexact iterates. Theorems B.11 and B.12 are local versions of Grazzi et al. (2020, Proposition 2.1 and Theorem 2.1), which assume a global contraction in the Euclidean norm. Here the contraction holds on an invariant ball in a weighted norm.

Lemma B.13 (Sufficient readout constants for mean cross-entropy)**.** *In this lemma and its proof, $R(C) = (R_t(C_{<t}))_{t\in\mathcal{I}} \in \mathbb{R}^{|\mathcal{I}|\times V_{\mathrm{voc}}}$ denotes the source logits at the scored positions $\mathcal{I}$ of Eq. (26), the only logits on which h depends. We equip these logit arrays with $\|z\|_{\max} = \max_{t\in\mathcal{I},v}|z_{t,v}|$, whose dual norm is the sum of absolute entries, and write $\|\cdot\|_{\max\leftarrow w}$ for the induced norm of linear maps from caches to logit arrays. If $R(\cdot)$ is continuously differentiable on an open neighborhood of $\mathcal{B}_R$ and, for all $C, C' \in \mathcal{B}_R$,*

$$\|DR(C)\|_{\max\leftarrow w} \leq M_R, \qquad \|DR(C) - DR(C')\|_{\max\leftarrow w} \leq L_{DR}\|C - C'\|_w, \tag{37}$$

then Eq. (27) holds with $M_h = 2M_R$ and $L_h = M_R^2 + 2L_{DR}$.

Proof. Write $h = \phi \circ R$, with ϕ the mean cross-entropy on logits. Its derivative for position t is $|\mathcal{I}|^{-1}(p_t - e_{y_t})$, where p_t is the softmax and e_{y_t} the target indicator. Consequently $\|D\phi(z)\|_{\max,*} = |\mathcal{I}|^{-1}\sum_{t\in\mathcal{I}}\|p_t - e_{y_t}\|_1 \leq 2$. The softmax bound proved in Lemma B.2 also gives $\|D\phi(z) - D\phi(z')\|_{\max,*} \leq |\mathcal{I}|^{-1}\sum_{t\in\mathcal{I}}\|z_t - z'_t\|_\infty \leq \|z - z'\|_{\max}$. Integrating DR along a cache segment gives $\|R(C) - R(C')\|_{\max} \leq M_R\|C - C'\|_w$. The chain rule gives $\|Dh(C)\|_{w,*} \leq 2M_R$. Splitting the difference as

$$\begin{aligned} Dh(C) - Dh(C') = {} & [D\phi(R(C)) - D\phi(R(C'))]DR(C) \\ & + D\phi(R(C'))[DR(C) - DR(C')] \end{aligned}$$

then gives $(M_R^2 + 2L_{DR})\|C - C'\|_w$. These cross-entropy constants require no bounded-logit assumption. □

The constants A_{g} and B_{g} in Eq. (35) are independent of the sequence length T when the contraction, initialization-error, derivative, and readout bounds hold uniformly over the lengths and inputs considered. Theorem B.6 supplies the required initialization-error bound under uniform basin conditions.

For a fixed finite sequence, Lemma 1 gives $C^{(T)}(\theta) = C^\star(\theta)$. If the computations are well defined and continuously differentiable on a parameter neighborhood, this identity holds throughout that neighborhood. Differentiating it gives $\nabla_\theta \mathcal{F}_T(\theta) = \nabla_\theta \mathcal{F}_\star(\theta)$ in exact arithmetic. This argument does not require contraction.

B.3 TRAINING THE MIXER WITH SGD

Section 1 asks how close a model trained through a few parallel rounds is to one trained through the exact recursion. This appendix answers it for a mixer trained with SGD, in the sense of stationarity. With a fixed number of rounds, SGD reaches an approximate stationary point of the sequential objective, up to an error of order K^2q^{2K}, where $q < 1$ is the contraction factor of Appendix B.2. Corollary B.15 combines the gradient bound of Appendix B.2 at a fixed θ with the analysis of SGD with biased gradients (Ajalloeian & Stich, 2020). Table 2 compares mixers trained through both methods empirically.

Dataset objective and sampling. For a finite dataset of N_{data} sequences, let $\mathcal{F}_{r,k}$ and $\mathcal{F}_{r,\star}$ denote the per-sequence mean losses in Eq. (26), and define

$$\overline{\mathcal{F}}_k(\theta) = \tfrac{1}{N_{\mathrm{data}}} \textstyle\sum_{r=1}^{N_{\mathrm{data}}} \mathcal{F}_{r,k}(\theta), \qquad \overline{\mathcal{F}}_\star(\theta) = \tfrac{1}{N_{\mathrm{data}}} \textstyle\sum_{r=1}^{N_{\mathrm{data}}} \mathcal{F}_{r,\star}(\theta). \tag{38}$$

Suppose the hypotheses of Theorem B.12 hold for every sequence on a common open convex parameter region Θ, with common upper bounds $q < 1$, A_{g} and B_{g}. By the triangle inequality over the finite sum, the dataset objectives then satisfy

$$\|\nabla\overline{\mathcal{F}}_k(\theta) - \nabla\overline{\mathcal{F}}_\star(\theta)\|_2 \le \delta_k, \qquad \delta_k := A_{\mathrm{g}}q^k + B_{\mathrm{g}}kq^{k-1}, \quad k \ge 1. \tag{39}$$

Let θ_n denote the mixer parameters at training step n, let $\mu_n > 0$ be the SGD step size, and let $N \ge 1$ denote the number of training steps.

At training step n, choose a deterministic number of rounds $K_n \ge 1$ and sample a minibatch uniformly with replacement, independently of the past. Initialize each sampled sequence with its plain cache, hold θ_n fixed for the inner computation, and differentiate its K_n cache updates and source readout. If g_n is the minibatch-average gradient and $\mathcal{H}_n$ is the history before this sample, then $\mathbb{E}[g_n \mid \mathcal{H}_n] = \nabla\overline{\mathcal{F}}_{K_n}(\theta_n)$. Thus g_n is unbiased for the finite-round objective and biased relative to $\overline{\mathcal{F}}_\star$. With $b_n = \nabla\overline{\mathcal{F}}_{K_n}(\theta_n) - \nabla\overline{\mathcal{F}}_\star(\theta_n)$ and $\xi_{n+1} = g_n - \nabla\overline{\mathcal{F}}_{K_n}(\theta_n)$, ordinary SGD has the decomposition

$$\theta_{n+1} = \theta_n - \mu_n g_n = \theta_n - \mu_n\big(\nabla\overline{\mathcal{F}}_\star(\theta_n) + b_n + \xi_{n+1}\big), \qquad \|b_n\|_2 \le \delta_{K_n}, \quad \mathbb{E}[\xi_{n+1} \mid \mathcal{H}_n] = 0. \tag{40}$$

Assumption B.14 (Conditions for SGD)**.** *Consider the sampling procedure described above and the SGD recursion in Eq. (40). The gradient-error bound Eq. (39) holds throughout the common open convex parameter region Θ. The following conditions also hold:*

1. *The initial parameter $\theta_0 \in \Theta$ is deterministic, and the iterates θ_n remain in Θ almost surely.*
2. *The exact dataset objective $\overline{\mathcal{F}}_\star$ has an $L_{\mathcal{F}}$-Lipschitz gradient on Θ, where $L_{\mathcal{F}} > 0$, and satisfies $\overline{\mathcal{F}}_\star(\theta) \ge \overline{\mathcal{F}}_{\inf}$ for every $\theta \in \Theta$, with $\overline{\mathcal{F}}_{\inf} \in \mathbb{R}$.*
3. *There is a finite constant $\sigma^2 \ge 0$ such that, at every training step, $\mathbb{E}[\|\xi_{n+1}\|_2^2 \mid \mathcal{H}_n] \le \sigma^2$. All expectations used in Corollary B.15 and its proof are finite.*
4. *The step sizes are deterministic and satisfy $0 < \mu_n \le 1/L_{\mathcal{F}}$.*

Remark (Assumptions for SGD). A Lipschitz objective gradient and bounded gradient-noise variance are standard conditions in convergence analyses of nonconvex stochastic optimization (Ghadimi & Lan, 2013). The requirement that θ_n remains in Θ ensures that the local bounds continue to apply throughout training. This is an additional stability assumption, which does not follow from contraction of the cache iteration.

Corollary B.15 (Expected stationarity for finite-round SGD)**.** *Under Assumption B.14, set $S_N = \sum_{n=0}^{N-1} \mu_n$ and $\Delta_{\mathcal{F}} = \overline{\mathcal{F}}_\star(\theta_0) - \overline{\mathcal{F}}_{\inf}$. Then*

$$\begin{aligned} \tfrac{1}{S_N}\textstyle\sum_{n=0}^{N-1} \mu_n \mathbb{E}\|\nabla\overline{\mathcal{F}}_\star(\theta_n)\|_2^2 \le{}& \tfrac{2\Delta_{\mathcal{F}}}{S_N} \\ &+ \tfrac{L_{\mathcal{F}}\sigma^2}{S_N}\textstyle\sum_{n=0}^{N-1} \mu_n^2 \\ &+ \tfrac{1}{S_N}\textstyle\sum_{n=0}^{N-1} \mu_n \delta_{K_n}^2. \end{aligned} \tag{41}$$

Proof. Put $a_n = \nabla\overline{\mathcal{F}}_\star(\theta_n)$. Convexity of Θ and the assumed parameter stability keep the update segment inside the smoothness region. Applying the descent inequality and conditioning on $\mathcal{H}_n$ gives

$$\mathbb{E}[\overline{\mathcal{F}}_\star(\theta_{n+1}) \mid \mathcal{H}_n] \le \overline{\mathcal{F}}_\star(\theta_n) - \mu_n\langle a_n, a_n + b_n\rangle + \tfrac{L_{\mathcal{F}}\mu_n^2}{2}\big(\|a_n + b_n\|_2^2 + \sigma^2\big).$$

Since $L_{\mathcal{F}}\mu_n \le 1$ and $2\langle a, a+b\rangle = \|a\|_2^2 + \|a+b\|_2^2 - \|b\|_2^2$,

$$-\mu_n\langle a_n, a_n + b_n\rangle + \tfrac{L_{\mathcal{F}}\mu_n^2}{2}\|a_n + b_n\|_2^2 \le -\tfrac{\mu_n}{2}\|a_n\|_2^2 + \tfrac{\mu_n}{2}\|b_n\|_2^2.$$

Use $\|b_n\|_2 \le \delta_{K_n}$, take expectations, and sum over $n = 0, \dots, N-1$. The objective differences telescope, and $\mathbb{E}\overline{\mathcal{F}}_\star(\theta_N) \ge \overline{\mathcal{F}}_{\inf}$ gives

$$\tfrac{1}{2}\textstyle\sum_{n=0}^{N-1}\mu_n\mathbb{E}\|a_n\|_2^2 \le \Delta_{\mathcal{F}} + \tfrac{1}{2}\sum_{n=0}^{N-1}\mu_n\delta_{K_n}^2 + \tfrac{L_{\mathcal{F}}\sigma^2}{2}\sum_{n=0}^{N-1}\mu_n^2.$$

Division by $S_N/2$ proves the claim. □

Stationarity and the number of rounds. For a prescribed training horizon N, take a constant step size $\mu_n = c/\sqrt{N}$, where $0 < c \le 1/L_{\mathcal{F}}$, and a fixed number of rounds $K \ge 1$. Equation (41) becomes

$$\tfrac{1}{N}\textstyle\sum_{n=0}^{N-1}\mathbb{E}\|\nabla\overline{\mathcal{F}}_\star(\theta_n)\|_2^2 \le \big(\frac{2\Delta_{\mathcal{F}}}{c} + L_{\mathcal{F}}c\sigma^2\big)N^{-1/2} + \delta_K^2. \tag{42}$$

The first term is the usual stochastic optimization bound. The second is the additional stationarity error of finite unrolling, with $\delta_K^2 = O(K^2q^{2K})$. A fixed number of rounds is therefore enough once δ_K^2 falls below the optimization error of the training budget. For fixed $q \in (0,1)$ and constants uniform in the horizon, the choice $K = \lceil c_0 \log N\rceil$ with $c_0 > 1/(4|\log q|)$ and $N \ge 2$ gives $\delta_K^2 = O((1+\log^2 N)N^{-2c_0|\log q|}) = O(N^{-1/2})$. A number of rounds that grows logarithmically with N thus keeps this error at the order of the first term. This analysis covers plain SGD on per-sequence mean losses. It does not cover the AdamW optimizer, gradient clipping, or the token mean over a minibatch used in our experiments.

B.4 Sharper convergence bounds

This appendix replaces the condition $\varepsilon\eta < 1$ of Theorem B.4 by the weaker condition $\chi < 1$ on the feedback gain of Section 3.3. Since χ grows with the mixing strength and the loop length, this condition explains why weaker mixing and shorter loops converge faster in Figure 3. Appendix B.4.1 bounds how much a difference is amplified as it passes through the recirculation loop, and Appendix B.4.2 shows when this bound guarantees convergence from the plain cache. We keep Assumption B.1, fix $0 \le \varepsilon \le \bar{\varepsilon}$ throughout, and write $C^\star$ for $C_\varepsilon^\star$.

B.4.1 The scalar feedback gain

Recall $m = L - d - 1$, the loop length $p = s - d$, and the nonnegative quantities A, B, g, v in Eq. (9). Put

$$\gamma_i = a_i + b_i\kappa_i, \qquad \widehat{g}_i = \kappa_i\textstyle\prod_{j=1}^{i-1}\gamma_j, \qquad M_\varepsilon = \varepsilon(I-A)^{-1}B. \tag{43}$$

This $\widehat{g}$ is the vector of Lemma B.5. We call M_ε the comparison matrix and write spr for the spectral radius. Lemma B.3 gives the componentwise comparison $e^+ \le M_\varepsilon e$.

Lemma B.16 (Rank-one comparison matrix and feedback gain)**.** *For nonnegative layer and mixer constants,*

$$\begin{aligned}(I-A)^{-1}g &= \widehat{g}, \qquad M_\varepsilon = \varepsilon\lambda\,\widehat{g}v^\top,\\ \chi := \mathrm{spr}(M_\varepsilon) &= \varepsilon\lambda\, v^\top\widehat{g} = \varepsilon\lambda\left[\textstyle\prod_{i=1}^p(a_i + b_i\kappa_i) - \prod_{i=1}^p a_i\right].\end{aligned} \tag{44}$$

Proof. For every integer $n \ge 1$, replacing factors one at a time gives

$$\textstyle\prod_{j=1}^n\gamma_j - \prod_{j=1}^n a_j = \sum_{r=1}^n(\gamma_r - a_r)\left(\prod_{j<r}\gamma_j\right)\left(\prod_{j=r+1}^n a_j\right). \tag{45}$$

Indeed, the r-th summand is the product with its first r factors replaced by γ_j minus the product with its first $r-1$ factors replaced, so the sum telescopes.

Using $\gamma_r - a_r = b_r\kappa_r$ and taking $n = i - 1$,

$$(A\widehat{g})_i = \kappa_i \sum_{r<i} b_r\kappa_r \left(\prod_{j<r} \gamma_j\right)\left(\prod_{j=r+1}^{i-1} a_j\right)$$
$$= \kappa_i \left(\prod_{j<i} \gamma_j - \prod_{j<i} a_j\right) = \widehat{g}_i - g_i.$$

The identity also holds at $i = 1$, where the sum is empty. Since $A^m = 0$, $I - A$ is invertible, so $\widehat{g} = (I - A)^{-1}g$. The definition $B = \lambda g v^\top$ then gives the rank-one representation in Eq. (44). Finally, $v_r = 0$ for $r > p$, and Eq. (45) at $n = p$ yields

$$v^\top \widehat{g} = \sum_{r=1}^{p} b_r\kappa_r \left(\prod_{j<r} \gamma_j\right)\left(\prod_{j=r+1}^{p} a_j\right) = \prod_{i=1}^{p} \gamma_i - \prod_{i=1}^{p} a_i.$$

The rank-one matrix $\varepsilon\lambda\widehat{g}v^\top$ has the eigenvalue $\varepsilon\lambda v^\top\widehat{g} \geq 0$ and otherwise only zero eigenvalues, which proves the spectral-radius formula. □

B.4.2 Convergence from the plain cache

Theorem B.17 establishes contraction when $\chi < 1$ and gives conditions ensuring that iteration from C^{pl} stays in the chosen neighborhood and converges to $C^\star$. Corollary B.18 then sharpens the error estimate for this initialization. The bound decreases by a factor χ with each additional round.

Theorem B.17 (Contraction under the scalar feedback condition)**.** *Suppose Assumption B.1 holds, fix $0 \leq \varepsilon \leq \bar{\varepsilon}$, and assume $\chi < 1$. Choose any $\rho \in (\chi, 1)$, and define*

$$\zeta = v^\top \mathbf{1}, \qquad \omega = \mathbf{1} + \tfrac{\varepsilon\lambda\zeta}{\rho-\chi}\widehat{g}, \qquad \|C\|_\omega = \max_i \tfrac{\|\mathsf{C}_i\|_{\mathrm{tok}}}{\omega_i}. \tag{46}$$

Then $\omega_i \geq 1$ and

$$\|\widehat{G}_\theta(C) - \widehat{G}_\theta(C')\|_\omega \leq \rho\|C - C'\|_\omega \quad (C, C' \in \mathcal{U}). \tag{47}$$

The condition of Proposition 1 then holds with rate ρ on every $\|\cdot\|_\omega$-ball about a fixed point contained in $\mathcal{U}$, by the argument of Appendix B.1.4 with ω in place of w. Moreover, $\chi < 1$ is necessary and sufficient for the comparison matrix *M_ε to admit a positive weighted maximum norm with induced norm less than one. Hence $\chi < 1$ is the weakest condition under which this comparison argument certifies contraction in a weighted maximum norm.*

For a plain-cache initialization, suppose Assumption B.1 holds on an open convex neighborhood of $\mathcal{B}_{\omega,R} = \{C : \|C - C^{\mathrm{pl}}\|_\omega \leq R\}$, where $R > 0$. Let h_0 be as in Lemma B.5, and set $D_\omega = h_0 \max_i \widehat{g}_i/\omega_i$. If $\varepsilon D_\omega \leq (1 - \rho)R$, then this ball is invariant and contains the unique fixed point $C^\star$, equal to the cache of sequential recirculation. In particular,

$$\|C^\star - C^{\mathrm{pl}}\|_\omega \leq \tfrac{\varepsilon D_\omega}{1-\rho}, \qquad \|C^{(k)} - C^\star\|_\omega \leq \rho^k \tfrac{\varepsilon D_\omega}{1-\rho} \quad (C^{(0)} = C^{\mathrm{pl}}). \tag{48}$$

Proof. Write $c = \varepsilon\lambda$, so that $M_\varepsilon = c\widehat{g}v^\top$ and $\chi = cv^\top\widehat{g}$. Direct multiplication gives

$$M_\varepsilon\omega = c\widehat{g}\left(\zeta + \tfrac{c\zeta}{\rho-\chi}v^\top\widehat{g}\right) = \tfrac{\rho c\zeta}{\rho-\chi}\widehat{g} = \rho(\omega - \mathbf{1}) \leq \rho\omega.$$

If e is the vector of input-cache differences, then $e \leq \omega\|C - C'\|_\omega$. Nonnegativity and the gain lemma imply $e^+ \leq M_\varepsilon\omega\|C - C'\|_\omega \leq \rho\omega\|C - C'\|_\omega$. Division by ω_i proves Eq. (47). It also proves sufficiency for contraction of the comparison matrix itself.

Conversely, suppose positive weights $\widetilde{\omega}$ satisfy $M_\varepsilon\widetilde{\omega} \leq r\widetilde{\omega}$ for some $r < 1$. This is the weighted row-sum characterization of its induced maximum norm. If $v = 0$, then $\chi = 0$. Otherwise $v^\top\widetilde{\omega} > 0$, and multiplication by $v^\top$ gives $\chi v^\top\widetilde{\omega} \leq rv^\top\widetilde{\omega}$, hence $\chi \leq r < 1$. This proves necessity for the comparison matrix.

The proof of Lemma B.5 gives the componentwise defect $\|[\widehat{G}_\theta(C^{\mathrm{pl}})]_i - \mathsf{C}_i^{\mathrm{pl}}\|_{\mathrm{tok}} \leq \varepsilon h_0\widehat{g}_i$, independently of the choice of weights. Thus the weighted defect is at most εD_ω. For $C \in \mathcal{B}_{\omega,R}$,

$$\|\widehat{G}_\theta(C) - C^{\mathrm{pl}}\|_\omega \leq \rho\|C - C^{\mathrm{pl}}\|_\omega + \varepsilon D_\omega \leq \rho R + \varepsilon D_\omega \leq R.$$

This proves invariance. The remaining claims follow as in the proof of Theorem B.6, with w, $\widehat{D}$ and $\widehat{q}_\varepsilon$ replaced by ω, D_ω and ρ. □

Comparison with the earlier contraction condition. Theorem B.4 requires $\varepsilon\eta < 1$. Its proof shows that $M_\varepsilon w \leq \varepsilon\eta w$ for the weights in Eq. (12), which implies $\chi \leq \varepsilon\eta$ by the converse part

of the proof of Theorem B.17. Thus the new condition $\chi < 1$ holds whenever the earlier condition holds, and can also hold when the earlier condition fails. For example, take $m = 2$, $p = 1$, $a_1 = b_1 = \kappa_1 = \lambda = 1$, and $\kappa_2 = 10$, with a_2 and b_2 arbitrary. Then $\eta = 10$ and $\chi = \varepsilon$. At $\varepsilon = 1/2$, we have $\varepsilon\eta = 5$ but $\chi = 1/2$, so only the new contraction test passes. This example uses illustrative layer bounds. Unlike η, χ does not depend on the layers above the source layer, which carry no feedback. Convergence from the plain cache still requires the local bounds to hold on the chosen neighborhood and the invariance condition $\varepsilon D_\omega \le (1-\rho)R$. Since the two results use different weighted balls, the weaker contraction condition does not automatically give a larger region of valid initializations.

Corollary B.18 (A sharper rate from the plain-cache defect)**.** *Under the invariant-ball hypotheses of Theorem B.17, the plain-initialized iterates satisfy, for every layer i and $k \ge 0$,*

$$\|\mathsf{C}_i^{(k)} - \mathsf{C}_i^\star\|_{\mathrm{tok}} \le \tfrac{\varepsilon h_0 \widehat{g}_i}{1-\chi}\chi^k, \qquad \|C^{(k)} - C^\star\|_\omega \le \tfrac{\varepsilon D_\omega}{1-\chi}\chi^k. \tag{49}$$

Here $\chi^0 = 1$, also when $\chi = 0$.

Proof. Let $e_i^\star = \|\mathsf{C}_i^\star - \mathsf{C}_i^{\mathrm{pl}}\|_{\mathrm{tok}}$. Comparing the fixed point with the image of the plain cache and adding the defect gives $e^\star \le M_\varepsilon e^\star + \varepsilon h_0 \widehat{g}$. Since $\mathrm{spr}(M_\varepsilon) = \chi < 1$, its nonnegative inverse is $(I - M_\varepsilon)^{-1} = I + M_\varepsilon/(1-\chi)$. Multiplying the componentwise inequality by this inverse, and using $M_\varepsilon \widehat{g} = \chi \widehat{g}$, yields

$$e^\star \le \varepsilon h_0 (I - M_\varepsilon)^{-1} \widehat{g} = \tfrac{\varepsilon h_0}{1-\chi}\widehat{g}.$$

All iterates stay in the certified ball, so their layer error vectors obey $e^{(k)} \le M_\varepsilon^k e^{(0)}$. Since $e^{(0)} = e^\star$, $M_\varepsilon \ge 0$ and $M_\varepsilon \widehat{g} = \chi \widehat{g}$, $e^{(k)} \le \varepsilon h_0 \chi^k \widehat{g}/(1-\chi)$. The weighted estimate follows by division and maximization. □

C Experimental setup

This appendix gives the details needed to reproduce the experiments of Sections 3.4 and 4. We first describe the models and hardware (Appendix C.1) and the implementations of the three computations (Appendix C.2). We then give the recirculation settings and how they were chosen (Appendix C.3), the data and tasks (Appendix C.4), the metrics and rounding scales (Appendix C.5), and mixer training (Appendix C.6). Finally, we describe the timing protocol (Appendix C.7).

C.1 Models, precision, and hardware

Models. Our main experiments use Gemma 3 (Gemma Team et al., 2025). We use the pretrained checkpoints 1B, 4B, and 12B for all experiments except the instruction task, which uses the instruction-tuned checkpoints 4B-IT and 12B-IT. The 4B and 12B checkpoints are multimodal, and we use only their language model. To test other architectures, we use Llama 3.1 8B (Grattafiori et al., 2024) and Qwen3 8B (Yang et al., 2025). For the instruction task, these are Llama 3.1 8B Instruct and Qwen3 8B in non-thinking mode, and for all other experiments the pretrained Llama 3.1 8B and Qwen3 8B Base, except the prefill timings, which use the instruction-tuned checkpoints of the same architecture. Table 5 lists the architectures. In Gemma 3, every sixth layer attends globally, and the other five attend to a sliding window of W positions. Llama and Qwen attend globally in every layer. Qwen3 has no beginning-of-sequence (BOS) token, so its inputs start directly with text. All weights are the public releases on the Hugging Face hub.

Table 5: Architecture details. L layers, hidden size H, number of query and key–value heads, head dimension, key or value dimension H_{kv} (KV heads × head dimension), number of global-attention layers, sliding window W, and vocabulary size.

Model	L	H	heads / KV	head dim	H_{kv}	global layers	W	vocabulary
Gemma 3 1B	26	1,152	4 / 1	256	256	4	512	262,144
Gemma 3 4B	34	2,560	8 / 4	256	1,024	5	1,024	262,208
Gemma 3 12B	48	3,840	16 / 8	256	2,048	8	1,024	262,208
Llama 3.1 8B	32	4,096	32 / 8	128	1,024	32	—	128,256
Qwen3 8B	36	4,096	32 / 8	128	1,024	36	—	151,936

Precision. Unless stated otherwise, weights and matrix products are in bf16, and the attention softmax and the mixing step of Section 2, including the norm matching, are computed in fp32. The LM head runs in bf16, and log-probabilities and all reported negative log-likelihood (NLL) and KL values are computed in fp32 from its output, or in fp64 in the convergence analysis. Where stated, the convergence analysis, the contraction estimates, and the gradient check run the entire model in fp32, with explicit attention in place of the FlashAttention kernel (Dao et al., 2022). Rounding scales are always measured at the precision of the experiment they accompany.

Hardware and software. All experiments except those of Appendix F.3 run on AMD Instinct MI210 GPUs with 64 GB of memory, in nodes of eight. Each evaluation and each timing process runs on one device, and large evaluations are split into independent shards over several devices. The exact trainer, the training timings, and the timings of Table 23 use data parallelism within one node. The roofline model of Section 4.2 uses the peak rates of the MI210, 181 TFLOP/s in bf16 and 1.6 TB/s. We use ROCm 6.4, PyTorch 2.9.1, transformers 4.57.3, flash-attention 2.8.3 (ROCm build), and Triton 3.5.1. The prefill timings of Appendix F.3 run on one NVIDIA A100 80GB PCIe and one NVIDIA H100 80GB PCIe GPU, each in a cloud virtual machine, with CUDA 12.8 and the same versions of PyTorch, transformers, flash-attention, and Triton.

C.2 Implementations

We compare three computations, the plain forward pass without recirculation, sequential recirculation, and Jacobi recirculation. Each has an optimized implementation, which we use for all timings.

Sequential recirculation. After a batched pass over layers $0, \dots, d$, each step runs the recirculated pass of token $t-1$ and the source pass of token t together as a two-row batch through layers $d+1, \dots, L-1$. The cache is preallocated, so every step has static shapes, and the step is compiled and captured once as a CUDA graph that is replayed $T-1$ times. Sliding-window layers keep only the last $W+1$ columns. The two-row attention is computed explicitly over the attended columns rather than with the FlashAttention decoding kernel, which on the MI210 is slower for two-row queries and slows down linearly with the cache length. On the A100 and H100, we time both and report the faster (Appendix F.3). Global layers read only the columns written so far, in buckets of 1,024 columns with one CUDA graph each. For 12B at 8k tokens and the largest batch that fits, we use buckets of 4,096 columns, since the compiled graphs of the smaller buckets do not fit in memory.

Jacobi recirculation. One round is the batched recirculated pass over layers $d+1, \dots, r_k$, which writes its keys and values into the cache in place, followed by the batched source pass over the same layers, which reads the rewritten cache at earlier positions together with its own fresh keys and values. Here $r_k = s$ in intermediate rounds and $L-1$ in the final round (Algorithm 1). Both passes use the FlashAttention kernel. The source pass combines the kernel's output over the rewritten columns with its own-column term through a log-sum-exp merge. Memory is that of an ordinary full-sequence forward pass plus the two full-length fp32 states z_d and z_s. For timing on a fixed input shape, the whole prefill is compiled and captured. In the complete evaluations, prompts have many different lengths, so we compile each layer separately and capture one graph per prompt shape.

Plain forward pass. A standard full-sequence forward pass with the same kernel, compilation, and graph capture.

Implementations per experiment. All timings of Section 4.2 use the optimized implementations, and compilation and capture are excluded from the times. In the perplexity evaluations, Jacobi recirculation and the plain pass are compiled but not captured, which changes their time by less than 0.5% and by 3–14%, respectively. On Llama 3.1 8B, compiling sequential recirculation slows the GSM8K prompt phase by 4%, so we report it with graph capture only. Token-by-token decoding on GSM8K uses the same uncompiled loop for all three computations.

C.3 Recirculation settings

Recirculation is specified by the source layer s, the destination layer d, the mixing coefficient $\alpha_\theta = \alpha$, and the mixing rule, either convex ($\beta_\theta = 1-\alpha$) or additive ($\beta_\theta = 1$). Table 6 lists the settings of each experiment. We take them from Mozer et al. (2026) where available and select them ourselves otherwise.

Settings from prior work. For the convergence analysis, the training-free perplexity evaluation, and the arXiv mixers, we use the settings for inputs with BOS provided by Mozer et al. (2026). For GSM8K, we use the 4B setting of their Appendix D.4. For 12B, we use their 12B perplexity pair for inputs without BOS. The GSM8K mixers keep only the layer pair, since the mixer produces its own coefficients. For the instruction task, we keep the layer pairs they selected for this task and choose α by the accuracy of sequential recirculation on the 800 questions, over $\{0.07, 0.10, 0.15\}$ for 4B-IT and $\{0.03, 0.05, 0.07, 0.10\}$ for 12B-IT. For this selection, the word order alternates between pairs, and the prediction is the most probable single candidate token. The held-out word list in Table 1 checks that the chosen coefficients do not overfit the original words.

Llama 3.1 8B and Qwen3 8B. For perplexity, we place the layer pair at the same relative depths as the Gemma 3 12B pair and use $\alpha = 0.07$ with the convex rule. The timings use the same pairs with $\alpha = 0.15$ and the additive rule. Neither choice affects the running time. For the instruction task, we evaluate sequential recirculation on every pair with $d \geq 1$ and loop length $p \leq 12$, 294 pairs on Llama and 342 on Qwen, at $\alpha = 0.15$ and $\alpha = 0.5$ with the additive rule. For each coefficient, we select the pair with the highest accuracy on the 800 questions. Since selection and evaluation use the same questions, we check the selection on held-out questions. Over 20 random splits into two halves, the pair selected on one half improves accuracy on the other half over the model without recirculation by 7.5 to 25.0 points on average and by at least 5.0 points on every split. For the GSM8K mixers, we reserve 500 problems of the training split, train a mixer for 200 steps for each of 12 candidate pairs on Llama 3.1 8B and 13 on Qwen3 8B, and compare pass@1 on these problems (Table 7). We select $(13, 5)$ for both models. On Llama 3.1 8B, it improves pass@1 from 46.0% to 53.0%. On Qwen3 8B, it reaches 87.8%, below the 88.4% of the model without recirculation but above all other candidates. The final mixers are trained on the full training split, and the test set is not used for selection.

Layer-pair and coefficient sweeps. The sweeps of Section 4.1 use the first 128 arXiv training windows of Appendix C.4, so no test data enter the selection. The layer-pair sweep evaluates sequential recirculation and Jacobi recirculation with $K = 2$ on every pair with $1 \leq d < s \leq L - 1$, which gives 300, 528, 1,081, 465, and 595 pairs for 1B, 4B, 12B, Llama 3.1 8B, and Qwen3 8B, at the coefficient and rule of the perplexity setting of each model. The coefficient sweeps evaluate sequential recirculation and Jacobi recirculation with $K = 2$ and $K = 3$ at $\alpha \in \{0.02, 0.03, 0.05, 0.07, 0.1, 0.15, 0.2, 0.3, 0.5\}$ on nine pairs. These are the best pair of each layer-pair sweep and, where it differs, the perplexity pair of the model, namely $(10, 4)$ and $(24, 22)$ on 1B, $(15, 8)$ on 4B, $(22, 13)$ and $(26, 10)$ on 12B, $(15, 9)$ and $(12, 2)$ on Llama 3.1 8B, and $(17, 10)$ and $(18, 10)$ on Qwen3 8B. Both sweeps use batches of 32 windows, or 16 on 12B.

Table 6: Recirculation settings used in the paper.

Used in	Model	(s, d)	α	Rule
Convergence (Section 3.4), training-free perplexity, arXiv mixers	1B	$(10, 4)$	0.05	convex
	4B	$(15, 8)$	0.09	additive
	12B	$(22, 13)$	0.08	additive
GSM8K and SVAMP	4B	$(16, 5)$	mixer	—
	12B	$(35, 16)$	mixer	—
Instruction task	4B-IT	$(18, 8)$	0.15	additive
	12B-IT	$(29, 16)$	0.10	additive
Instruction task	Llama 3.1 8B	$(12, 10)$ / $(13, 8)$	0.15 / 0.5	additive
	Qwen3 8B	$(17, 11)$ / $(21, 10)$	0.15 / 0.5	additive
GSM8K and SVAMP	Llama 3.1 8B	$(13, 5)$	mixer	—
	Qwen3 8B	$(13, 5)$	mixer	—
Training-free perplexity	Llama 3.1 8B	$(15, 9)$	0.07	convex
	Qwen3 8B	$(17, 10)$	0.07	convex
Prefill timing	Llama 3.1 8B	$(15, 9)$	0.15	additive
	Qwen3 8B	$(17, 10)$	0.15	additive

Table 7: Layer-pair selection for the GSM8K mixers on Llama 3.1 8B and Qwen3 8B: pass@1 in % on 500 held-out problems of the training set after 200 training steps, evaluated with sequential recirculation.

Llama 3.1 8B		Qwen3 8B	
(s, d)	Pass@1	(s, d)	Pass@1
Base	46.0	Base	88.4
$(13, 5)$	**53.0**	$(13, 5)$	**87.8**
$(17, 5)$	50.4	$(20, 12)$	87.4
$(21, 5)$	49.2	$(32, 16)$	87.0
$(20, 8)$	49.0	$(17, 9)$	86.8
$(16, 8)$	47.2	$(24, 16)$	86.8
$(24, 8)$	45.2	$(24, 12)$	86.6
$(27, 11)$	43.8	$(25, 9)$	86.4
$(22, 14)$	43.6	$(26, 12)$	86.4
$(19, 11)$	43.0	$(28, 16)$	86.2
$(23, 11)$	40.4	$(28, 12)$	86.0
$(26, 14)$	40.4	$(17, 5)$	85.0
$(30, 14)$	40.4	$(21, 5)$	85.0
		$(21, 9)$	84.8

C.4 Data and tasks

Perplexity. We evaluate perplexity on eight long-document corpora and on LAMBADA (Table 8). Each document is tokenized as a whole, with BOS at its start, and cut into consecutive chunks of 1,024 tokens. The first chunk of a document already starts with BOS and forms one window. Every later chunk forms a window of BOS followed by its first 1,023 tokens. Every window thus starts with BOS as the model expects, and 1,023 tokens are scored. This differs from the original protocol of Mozer et al. (2026), in which only the first chunk of each document starts with BOS. Only complete chunks are used, except for Newsroom, whose articles are mostly shorter than 1,024 tokens. There, the last partial chunk of each article is kept if it has at least two tokens, and the padding is masked. For BigPatent, we use the first 10,000 of the 67,072 test documents. C4 has no public test split, so we use the first 25,000 documents of its English validation split, about 4M tokens, the scale of the C4 evaluation of Mozer et al. (2026). For LAMBADA, each of the 5,153 test passages forms one window, and every token of the passage is scored. This is a perplexity over passages, not the usual accuracy on the last word. Perplexity is token-weighted over all windows of a dataset. Batch sizes depend on the model and the computation, and batch composition affects the results only at the rounding scale. On Llama 3.1 8B and Qwen3 8B, we evaluate the arXiv documents with each model's tokenizer, which gives 46,964 and 50,342 windows. Trained mixers are evaluated on arXiv, the dataset of their training. All datasets are loaded from the Hugging Face hub (`ccdv/arxiv-summarization`, `ccdv/pubmed-summarization`, `ccdv/govreport-summarization`, `FiscalNote/billsum`, `big_patent`, `emozilla/pg19-test`, `allenai/c4`, `EleutherAI/lambada_openai`), except Newsroom, which comes from its official release.

Table 8: Perplexity datasets. Documents counts the documents read, and Windows the windows evaluated.

Dataset	Split	Documents	Windows
arXiv (Cohan et al., 2018)	test	6,440	48,969
PubMed (Cohan et al., 2018)	test	6,658	21,894
GovReport (Huang et al., 2021)	test	973	8,511
BillSum (Kornilova & Eidelman, 2019)	test	3,269	5,696
BigPatent (Sharma et al., 2019)	test, first 10,000	10,000	67,048
PG-19 (Rae et al., 2020)	test	100	9,662
Newsroom (Grusky et al., 2018)	test	108,862	153,023
C4 (Raffel et al., 2020)	validation, first 25,000	25,000	4,101
LAMBADA (Paperno et al., 2016)	test	5,153	5,153

Training, convergence, and timing inputs. Mixer training and the sweeps use windows from the arXiv training split. We read the split in order, skip documents shorter than 2,000 characters, take at most two consecutive windows of BOS followed by 1,023 tokens from each document, and stop at 500 windows, which come from the first 261 documents. The sweeps use the first 128 of these windows. The convergence analysis uses the first 64 windows built in the same way from the test split, which come from the first 33 test documents. The length study and the prefill timings of Gemma 3 use 16 PG-19 test books, drawn at random from the 91 books with at least 16,384 tokens. Each input is BOS followed by the beginning of a book, truncated to the required length. For Llama 3.1 8B and Qwen3 8B, the prefill timings use the first 16 test books with at least 16,384 tokens under each model's tokenizer.

Instruction task. This task, from Appendix D.1 of Mozer et al. (2026), asks the model to apply a positional rule to a word pair. The model is given a fruit word and an animal word together with an instruction of the form "if the first word is a fruit, say first; if the second word is a fruit, say second", or the same instruction with animal in place of fruit. The 20 fruit words are apple, avocado, banana, blueberry, cantaloupe, cherry, grape, honeydew, kiwi, lemon, lime, mango, orange, peach, pear, pineapple, plum, raspberry, strawberry, and watermelon. The 20 animal words are bear, bird, cat, deer, dog, dolphin, elephant, fox, giraffe, lion, lizard, monkey, penguin, shark, snake, spider, tiger, whale, wolf, and zebra. Every fruit is paired with every animal, which gives 400 pairs, and each pair is asked once with the fruit instruction and once with the animal instruction, which gives 800 questions with balanced answers. On Gemma 3, each question is the following prompt in the chat template, preceded by BOS and without a system message. The fruit version is shown. The animal version replaces fruit with animal and swaps the answers of the two examples.

```
Let's play a game.  I will say two words.
If the first word is a fruit, you say first.
If the second word is a fruit, you say second.
For example, if I say 'pomegranate kangaroo', respond first.
And if I say 'koala fig', respond second.
The word pair is 'bear apple'.  Your answer?
```

We construct the questions following the description of Mozer et al. (2026) and write every pair animal first, as in their example 'monkey banana'. Since accuracy on this task depends on such details, we keep this construction for all Gemma 3 models. Appendix E.1 reports the other order. The answer is read from the logits at the first generated position. The candidates are the tokens of first and second in four surface forms each, with and without a leading space and in lower and upper case, and the prediction is the group with the larger total probability. The held-out word list uses 20 fruits (apricot, blackberry, coconut, cranberry, durian, grapefruit, guava, jackfruit, lychee, nectarine, papaya, persimmon, quince, tangerine, clementine, mulberry, gooseberry, starfruit, passionfruit, plantain) and 20 animals (alligator, antelope, badger, beaver, buffalo, camel, cheetah, chimpanzee, cobra, crocodile, donkey, eagle, falcon, gorilla, hamster, hedgehog, hippo, jaguar, leopard, otter) that appear neither in their lists nor among the example words of the prompt, with the same template and scoring. For Llama 3.1 8B Instruct and Qwen3 8B, we use each model's chat template with the system message "Answer with a single word: first or second. No explanation." Without it, Llama explains before answering, and the candidate tokens carry little probability. For these two models, the word order alternates between pairs.

GSM8K. We use the complete test set of 1,319 problems (Cobbe et al., 2021). Each question is presented zero-shot as

```
Q: {question}
A: Let's think step by step.
```

preceded by BOS for the models that have one. The model generates greedily for at most 320 new tokens or until its end-of-sequence token. Prompts of equal token length are batched together, so no padding is needed. Base models often continue with further questions after answering, so we cut the generation at the first line that starts a new question (`Q:`, `Question:`, `[Question]`, or `Problem:`). This cut affects most generations of the base 12B and Qwen3 8B models, but none of the mixers on 4B, Llama, and Qwen, whose generations end with "The answer is ..." and the end-of-sequence token. The predicted answer is the last number in the remaining text, and a problem counts as solved if it equals the gold answer numerically (pass@1). We evaluate two

computations, sequential recirculation for prefill and decoding, and Jacobi recirculation with K rounds for prefill followed by the same sequential decoding. Agreement is the fraction of problems on which the two computations give the same final answer. We evaluate 1B on GSM8K only for timing. With one mixer trained through Jacobi recirculation at the pair $(10, 4)$, its pass@1 is 1.4% without recirculation and 2.5% or 2.3% with the mixer evaluated through sequential recirculation or Jacobi recirculation with $K = 2$, which is too low to be informative.

SVAMP. We use all 1,000 problems of the training and test splits of SVAMP (Patel et al., 2021). Each question is the problem body followed by the question, presented with the prompt, decoding, and scoring of GSM8K. The GSM8K mixers trained through Jacobi recirculation are applied without further training.

C.5 Metrics and rounding scales

Convergence. The convergence analysis runs 12 rounds on the 64 test windows. Every round runs to the top layer, so the state after round k equals Jacobi recirculation with $K = k$. The reference is sequential recirculation in the same batches and precision. The cache error covers the keys and values of layers $d+1, \ldots, L-1$ at positions $0, \ldots, T-2$, since the recirculated pass never rewrites the last position, and the per-column error restricts it to one position. The prediction error averages the KL divergence of each window over positions $0, \ldots, T-2$, computed in fp64. We report the median over windows for the cache error and the mean for the prediction error. The length study uses one book per batch and four rounds. In fp32, it stops at 4k tokens on 12B, since longer prefixes do not fit on one device. The per-position analysis of Figure 10 uses 4k-token prefixes in fp32, of eight books on 4B and one book on 12B.

Contraction. The per-round contraction $\hat{\rho}$ of Figure 3(A, B) is computed in fp32 with 12 rounds. Let m_k be the median cache error over the 64 windows after round k. We take the ratios m_k/m_{k-1} for $k \geq 2$ with $m_{k-1} \geq 10^{-4}$, which excludes errors that have reached the fp32 rounding scale, and report their median. The coefficient sweep uses $\alpha \in \{0.02, 0.05, 0.1, 0.15, 0.2, 0.3, 0.5\}$ on 1B, $\{0.02, 0.05, 0.09, 0.15, 0.2, 0.3, 0.5\}$ on 4B, and $\{0.02, 0.04, 0.08, 0.15, 0.3, 0.5\}$ on 12B. The loop-length sweep keeps d and α and moves the source layer to $s \in \{6, 8, 10, 12, 16, 20\}$ on 1B, $\{10, 12, 15, 18, 22, 26\}$ on 4B, and $\{15, 17, 22, 26, 30, 36\}$ on 12B. The trained mixers are the seed-0 arXiv mixers of Section 4.1.

Rounding scales. Each rounding scale compares two runs of sequential recirculation on the same inputs that differ only in how the inputs are batched, with the metric of the experiment it accompanies. The convergence analysis compares batches of 4 and 1, and the layer-pair sweeps rerun sequential recirculation on every 12th to 48th pair.

Gradient check. Figure 4(C, D) compares the exact gradient of the training loss with respect to the mixer parameters with the gradient through K rounds. We evaluate both at steps $0, 20, \ldots, 100$ of the seed-0 arXiv mixers trained through Jacobi recirculation, on the first eight training windows, with the entire model in fp32. The exact gradient backpropagates through the token-by-token computation of sequential recirculation. Both gradients are computed for each window separately, and the figure reports the mean of the per-window cosines. Appendix E.4 discusses individual windows and the cosine of the batch gradient.

C.6 Mixer training

Mixer. The mixer follows the adaptive variant of Mozer et al. (2026). It takes the concatenation $[z_{s,t}; z_{d,t}] \in \mathbb{R}^{2H}$, applies a layer normalization and two hidden layers of width H with GELU activations, and outputs $2H$ values that a sigmoid maps to the per-dimension coefficients $\alpha_\theta, \beta_\theta \in (0, 1)^H$ of Section 2. The norm matching f is kept. The last layer is initialized to zero with a bias such that every mixer starts at $\alpha = 0.1$ and $\beta = 0.9$. The mixer runs in fp32. The transformer weights are frozen, and only the mixer is trained.

Recipe. All mixers are trained with AdamW at a constant learning rate of 3×10^{-4}, with the default momentum parameters, weight decay 10^{-4}, and gradient clipping at norm 1, on the mean NLL over the target tokens of each step. For perplexity, the training data are the 500 arXiv training windows of Appendix C.4. Each step draws 32 of them with replacement, all 1,023 targets of a window enter the loss, and a mixer is trained for 100 steps. For GSM8K, the training data are the 7,473 problems

of the training split. Each example is the evaluation prompt followed by the reference solution, whose final line is rewritten as "The answer is ..." and followed by the end-of-sequence token, truncated to 320 tokens. The loss covers the solution tokens only. Each step draws 4 problems with replacement, and a mixer is trained for 800 steps. We evaluate the final checkpoint. The seeds 0, 1, and 2 determine the draws and the initialization of the two hidden layers, which are the same for both training methods under a given seed.

Exact and Jacobi training. The Jacobi trainer runs Algorithm 1 with $K = 2$ in the forward pass and backpropagates through both rounds. The exact trainer backpropagates through sequential recirculation itself. It keeps the KV cache in preallocated buffers, recomputes the attention softmax of every layer and step in the backward pass, and runs the backward pass in reverse token order, so that the memory per step does not grow with T. The exact trainer uses data parallelism, with four windows on each of eight devices for the arXiv mixers, and one problem on each of four devices for the GSM8K mixers. Both trainers see the same examples in every step and start from the same mixer.

C.7 Timing protocol

Prefill. We time the plain forward pass, sequential recirculation, and Jacobi recirculation on the PG-19 inputs of Appendix C.4. A batch larger than 16 repeats the books. We vary one factor at a time from $T = 1{,}024$, $B = 1$, and $K = 2$, with T from 1k to 16k tokens, B from 1 to 32 (up to 64 on 4B and 128 on 1B), and K from 1 to 16. For Figure 5 (right) and Table 22, we time each model at $T \in \{1\text{k}, 4\text{k}, 8\text{k}, 16\text{k}\}$ at the largest batch at which sequential recirculation fits in memory, found by halving the batch until it fits. Each measurement runs as the only process on its node, to avoid interference between processes. Each measurement follows warm-up runs that include compilation and capture, three for the plain pass and Jacobi recirculation and one for sequential recirculation. It reports the median of five repetitions, with device synchronization before and after each run. Beyond 4k tokens, where a single run of sequential recirculation takes minutes, we report the slower of two repetitions. Repetitions differ by less than 2% in 96% of the measurements, measured as the range over the median. In the grid of Table 21, the spread stays below 5.3% for sequential recirculation and 4.0% for Jacobi recirculation, and at the largest batches of Table 22 below 11.3% for both. It is largest for the plain forward pass on 1B, whose runs take milliseconds, and reaches 19% there. We report times excluding the LM head, which stop after the last transformer layer and measure the cost of producing the hidden states and the KV cache, the prefill cost. Some tables also report times including the LM head and the log-softmax over the vocabulary at every position, as in a perplexity evaluation, which adds the same cost to all three computations.

Complete evaluations. The complete evaluations are timed with the optimized implementations on the data of Appendix C.4, at the batches listed in Table 4. Each prompt shape is compiled and captured once, and this time is excluded. Every batch is timed once. Perplexity times are GPU-hours summed over all shards, excluding the first batch of each shard. On GSM8K, the prompt phase ends when the prompt cache is complete, and the total time includes decoding. These evaluations run up to eight processes on a node, one per device. On 4B subsets of each scenario, running alone changes the times of sequential recirculation and Jacobi recirculation by less than 2%. The compiled prompt phase of sequential recirculation varies between runs, mainly with whether its compiled kernels are already cached. For the 4B-IT instruction task, it takes 18 to 28 s in repeated runs, whereas Base and Jacobi recirculation vary by less than 3%. Table 4 reports the fastest run, so its speed-up is conservative.

Training. Training times are measured on complete runs with the recipes of Appendix C.6, using optimized versions of both trainers. The exact trainer computes the backward pass of the frozen layers explicitly with static shapes and replays each token step as a CUDA graph. On short sequences in fp32, its gradients agree with those of the trainer of Appendix C.6 up to a relative error of 10^{-4}. The Jacobi trainer is compiled and runs on the same devices with the same batch per device. We report the sum of the step times from the second to the last step. Each step includes gradient synchronization and the optimizer update. The first step, which contains compilation, is excluded, as are checkpointing and model loading. The Llama 3.1 8B and Qwen3 8B runs on four devices shared one node. Timed alone for 20 steps, their step times change by less than 1%.

Memory. Peak memory is the maximum device memory allocated during a measured run, in GiB (2^{30} bytes) for prefill and in GB (10^9 bytes) for training. Prefill memory is measured without compilation.

FLOPs and bytes. FLOPs, bytes, and roofline bounds are not measured but counted analytically from the model geometry, as described in Appendix F.1.

D CONVERGENCE: SUPPLEMENTARY RESULTS

This appendix supplements Section 3.4. We show the convergence of every model, the same analysis in the other precision, the dependence on the sequence length and the position, the contraction estimates behind Figure 3, and how the rounding scale compares with the difference between two implementations. Appendix C.5 describes the measurements.

Per-model convergence. Figure 6 repeats Figure 2 with the per-column cache error and the length dependence shown for each model.

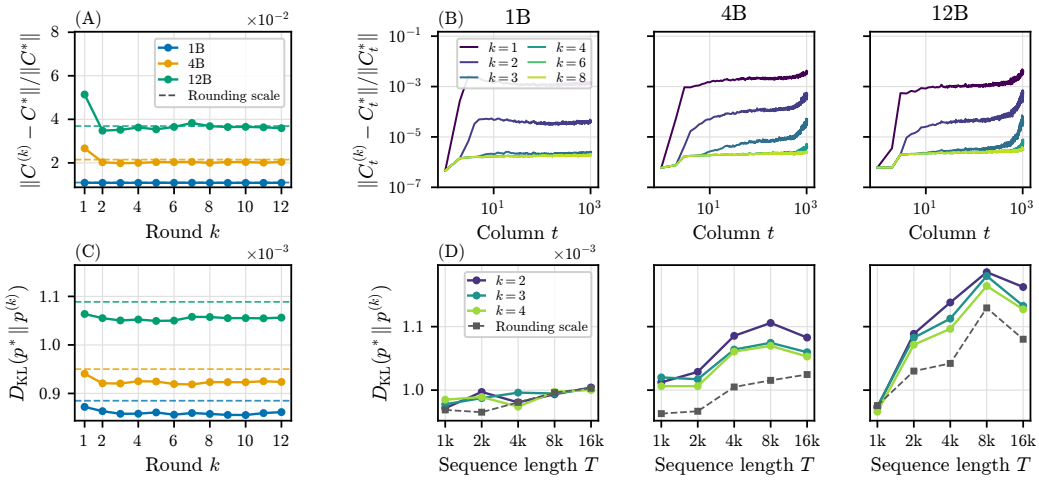


Figure 6: Empirical convergence on Gemma 3 1B, 4B, and 12B, with (B) and (D) for every model (the full version of Figure 2). (A) Overall cache error versus rounds in bf16. (B) Median per-column cache error after k rounds in fp32, where curves from round k onward overlap on the first k columns (Lemma 1). (C) Prediction error versus rounds in bf16. (D) Prediction error versus sequence length in bf16. Dashed lines mark the rounding scale.

Convergence in bf16. Table 9 divides the bf16 errors by the rounding scale. The prediction error reaches the scale after one round on all three models. The cache error reaches it after one round on 1B and after two rounds on 4B and 12B, where one round leaves it 24% and 39% above the scale. Ratios slightly below one arise because Jacobi recirculation and its reference run in the same batches, whereas the rounding scale compares two batch sizes. In bf16, the per-column error is nearly the same after every round (Figure 7), so bf16 rounding hides the column-by-column progress that fp32 reveals.

Convergence in fp32. In fp32 (Figure 8), each round reduces the median cache error to 3%–5% of its previous value on 1B, to 6%–16% on 4B, and to 7%–11% on 12B, until it levels off after four to six rounds. The level lies 1.6 to 2.3 times above the fp32 rounding scale. The prediction error follows the same pattern and levels off 1.8 to 7 times above its rounding scale, after four rounds on 1B and up to eight on 12B. Both levels match the difference between two implementations of sequential recirculation, as we show at the end of this appendix.

Sequence length and position. In bf16, the prediction error after two rounds stays within 10% of the same-length rounding scale for every model and length, and at most 9% above it (12B at 4k tokens). fp32 resolves errors far below this scale and gives a more detailed picture (Figure 9). On 1B, the median error after each round hardly depends on the length. On 4B and 12B, the error after

Table 9: bf16, T=1,024, 64 arXiv windows. Prediction error and overall cache error after k rounds, divided by the bf16 rounding scale.

	Prediction error / scale, $k = 1, 2, 3, 4$				Cache error / scale, $k = 1, 2, 3, 4$			
1B	0.99	0.98	0.97	0.97	0.99	0.99	0.98	0.99
4B	0.99	0.97	0.97	0.97	1.24	0.94	0.92	0.93
12B	0.98	0.97	0.96	0.97	1.39	0.94	0.95	0.98

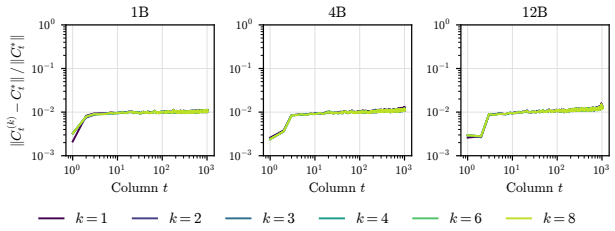


Figure 7: Median per-column cache error after k rounds in bf16 over the 64 windows. It barely changes between rounds, since bf16 rounding hides the progress visible in fp32.

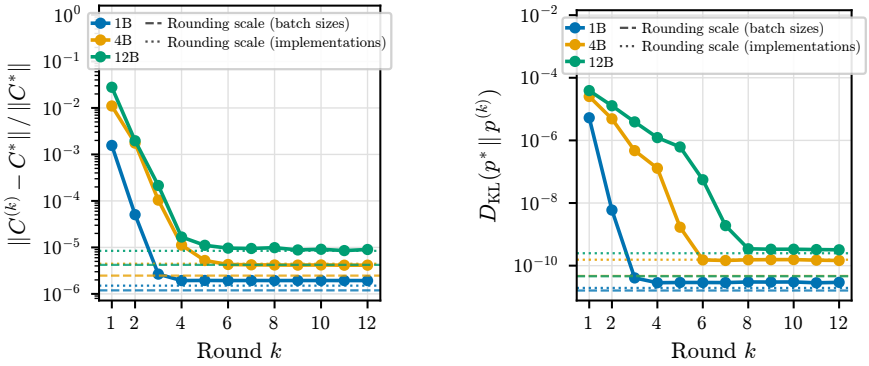


Figure 8: Overall cache error (left) and prediction error (right) versus rounds in fp32. Dashed lines show the fp32 rounding scale from two batch sizes, and dotted lines the difference between two implementations of sequential recirculation.

a fixed number of rounds grows with the length up to about 4k tokens, and on 4B it then grows much more slowly up to 16k tokens. Each round still reduces it, but by less than at 1k tokens. On 4B, for example, the median prediction error falls from 8×10^{-8} to 5×10^{-11} between two and four rounds at 1k tokens, but from 3×10^{-5} to 3×10^{-6} at 16k tokens. At these lengths, a few documents converge slowly and dominate the mean. Within a document, the error grows with the position, and on 4B additional rounds reduce it less at late positions than at early ones (Figure 10). These errors remain far below the bf16 rounding scale, so they do not affect the number of rounds needed in bf16. We leave the length dependence at higher precision to future work.

Mixing strength and loop length. Table 10 lists the contraction estimates of Figure 3(A, B). As α increases from 0.02 to 0.5, $\hat{\rho}$ grows from 0.015–0.036 to 0.30–0.61. As the loop length increases from 2 to 16–23 layers, it grows from 0.037–0.049 to 0.14–0.38. Over the same range of loop lengths, the prediction error after one round grows by two to three orders of magnitude and the cache error after one round by one to two. The trained arXiv mixers reach $\hat{\rho}$ between 0.22 and 0.39, close to fixed coefficients between 0.2 and 0.5, with no systematic difference between mixers trained through sequential recirculation and through Jacobi recirculation. For weak mixing and short loops, the error falls below 10^{-4} after two rounds, so $\hat{\rho}$ rests on a single ratio. We also ran $\alpha = 0.01$ on 1B and 4B, which Figure 3(A) omits. On 1B, no ratio qualifies there.

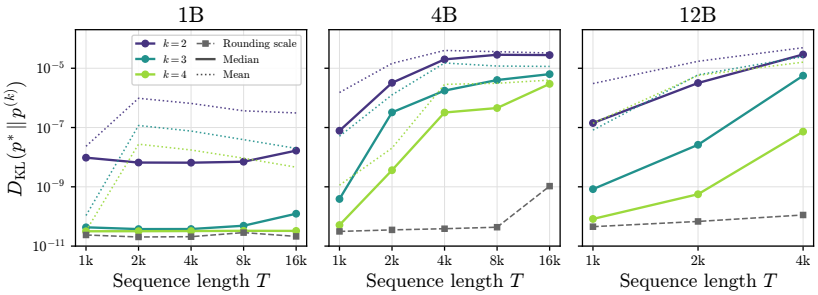


Figure 9: Prediction error versus sequence length for $k = 2, 3, 4$ rounds in fp32 on prefixes of 16 PG-19 documents. Solid and dotted lines show the median and mean over documents. Dashed lines show the same-length rounding scale. The 12B fp32 run does not fit on one device beyond 4k tokens.

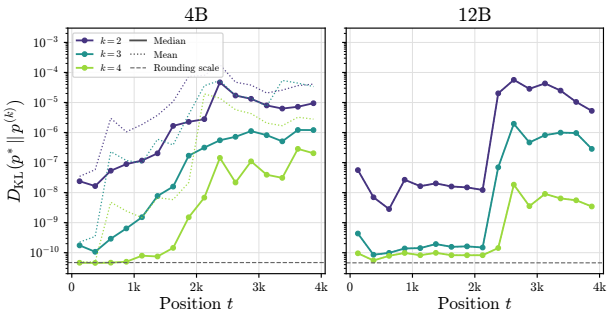


Figure 10: Prediction error per 256-position bucket on 4k-token PG-19 prefixes in fp32, with the median (solid) and mean (dotted) over eight documents on 4B and one document on 12B. The dashed line marks the fp32 rounding scale of the 64 arXiv windows. The error grows with the position, and on 4B additional rounds reduce it less at late positions than at early ones.

Table 10: Per-round contraction $\hat{\rho}$ in fp32 on 64 arXiv windows with 12 rounds, varying the mixing coefficient α at the layer pair of Table 6, or the loop length p at its destination layer and coefficient. The last block gives the seed-0 arXiv mixers.

1B	α	0.02	0.05	0.1	0.15	0.2	0.3	0.5
	$\hat{\rho}$	0.015	0.032	0.073	0.112	0.161	0.275	0.612
4B	α	0.02	0.05	0.09	0.15	0.2	0.3	0.5
	$\hat{\rho}$	0.036	0.064	0.108	0.155	0.170	0.256	0.299
12B	α	0.02	0.04	0.08	0.15	0.3	0.5	
	$\hat{\rho}$	0.021	0.060	0.078	0.167	0.273	0.393	
1B	p	2	4	6	8	12	16	
	$\hat{\rho}$	0.037	0.029	0.032	0.054	0.075	0.140	
4B	p	2	4	7	10	14	18	
	$\hat{\rho}$	0.043	0.093	0.108	0.124	0.182	0.184	
12B	p	2	4	9	13	17	23	
	$\hat{\rho}$	0.049	0.063	0.078	0.196	0.223	0.376	

Mixer	1B	4B	12B
Trained through SR	0.242	0.312	0.353
Trained through JR	0.222	0.299	0.385

Rounding scale and implementation. Throughout the paper, the rounding scale compares two batch sizes of the same implementation of sequential recirculation. Different implementations of the same computation also differ by rounding, since they use different kernels and orders of reduction. To check that the batch-split scale captures this, we compare the optimized implementation of sequential recirculation with the token-by-token reference on the 64 windows, and the plain forward

pass computed token by token with the batched full-sequence pass. In bf16, the two implementations of sequential recirculation differ by 0.97 to 1.07 times the batch-split scale, for both errors and all three models, and the two implementations of the plain pass by 0.4 to 0.9 times. In fp32, the implementations of sequential recirculation differ by up to 2.0 times the batch-split scale in the cache error and up to 5.4 times in the prediction error (12B), since fp32 resolves the different reduction orders. The median cache error at which Jacobi recirculation levels off in fp32, 1.9, 4.2, and 9.2×10^{-6} on 1B, 4B, and 12B, matches this cross-implementation difference of 1.5, 4.5, and 8.4×10^{-6}, and so does the prediction error, 2.9, 15, and 33×10^{-11} against 1.9, 15, and 25×10^{-11}. The batch-split scale is thus representative in bf16 and conservative in fp32.

E EXACTNESS: SUPPLEMENTARY RESULTS

This appendix supplements Section 4.1 in its order. We give further results on the instruction task, the complete perplexity results, the stability of hyperparameter selection, the gradient accuracy on individual windows, and further results on generation.

E.1 INSTRUCTION TASK

Other architectures. Table 11 repeats Table 1 on Llama 3.1 8B Instruct and Qwen3 8B, with the settings of Appendix C.3 and the prompt of Appendix C.4. For each model and coefficient, we report the pair with the highest accuracy of sequential recirculation. With two rounds, the agreement between Jacobi recirculation and sequential recirculation is at or above the rounding scale in three settings and 0.4 points below it in the fourth. With one round, it falls 6.5 points below the rounding scale on Llama 3.1 8B at $\alpha = 0.5$, so the stronger coefficient needs the second round, as on Gemma 3 4B-IT.

Table 11: Instruction task on Llama 3.1 8B Instruct and Qwen3 8B, 800 questions, accuracy in %. JR uses K rounds. Agreement is the fraction of identical SR and JR answers, and the rounding scale is the agreement between two runs of SR with different batch sizes.

Model, (s, d), α	Base	SR	JR K=1 / 2 / 3	Agreement K=1 / 2 / 3	Rounding scale
Llama 3.1 8B $(12, 10)$, 0.15	63.5	73.4	71.4 / 73.0 / 72.0	94.8 / 95.4 / 95.9	95.8
Llama 3.1 8B $(13, 8)$, 0.5	63.5	88.5	86.0 / 89.2 / 89.0	91.5 / 98.2 / 98.5	98.0
Qwen3 8B $(17, 11)$, 0.15	81.8	90.1	88.8 / 89.1 / 89.2	98.1 / 98.5 / 98.1	98.1
Qwen3 8B $(21, 10)$, 0.5	81.8	97.6	97.1 / 97.6 / 97.4	98.8 / 99.5 / 99.5	99.2

Word order and baselines. The order of the two words in a pair matters on this task. The models are strongly asymmetric on this task, so the baseline depends on the construction while the recirculated result does not (bf16, 400 questions per cell, in a separate diagnostic run). 12B-IT answers the "animal" instruction correctly on 62.8% of animal-first pairs and 100% of fruit-first pairs, and the "fruit" instruction on 100% and 97.5%. With $(29, 16)$ and α=0.07, the failing cell rises to 97.8%. 4B-IT answers 74.2% and 91.2% of animal-first pairs correctly under the fruit and animal instructions, and 62.0% and 26.8% of fruit-first pairs. The candidate tokens carry nearly all the probability, the margin between "first" and "second" is close to one on nearly every question, and bf16 and fp32 agree on 97.5% to 98.9% of the questions, so the baselines reflect systematic biases of the models rather than numerical noise. Animal-first pairs reproduce their 4B-IT baseline (82.9% against 82.3%) and, with $\alpha = 0.07$, both recirculated results (87.8% against 87.3%, and 98.5% against 98.4%). We use the animal-first construction on Gemma 3 and do not tune it to match individual reported numbers.

E.2 PERPLEXITY

Gemma 3. Table 12 lists the perplexities behind Figure 3(C).

Other architectures. On the complete arXiv test set, with the settings of Table 6, Llama 3.1 8B has a perplexity of 7.376 without recirculation, 7.375 with sequential recirculation, and 7.375 with Jacobi recirculation. On Qwen3 8B, the corresponding values are 6.486, 6.477, and 6.476. Recirculation thus changes perplexity only slightly on these models, and Jacobi recirculation reproduces sequential recirculation within 0.02%.

Table 12: Training-free perplexity on nine datasets (the validation split for C4), with JR using $K=2$. Base is the model without recirculation. Blue marks a perplexity below Base and red one above it.

	1B $(10, 4)$, α=0.05			4B $(15, 8)$, α=0.09			12B $(22, 13)$, α=0.08		
Dataset	Base ↓	SR ↓	JR ↓	Base ↓	SR ↓	JR ↓	Base ↓	SR ↓	JR ↓
arXiv	11.406	11.411	11.410	7.809	7.790	7.789	6.693	6.683	6.682
PubMed	10.614	10.621	10.620	7.424	7.413	7.411	6.449	6.443	6.442
BillSum	4.039	4.043	4.043	2.939	2.938	2.938	2.581	2.580	2.580
GovReport	9.306	9.309	9.309	7.113	7.102	7.101	6.124	6.115	6.115
BigPatent	6.669	6.674	6.674	5.061	5.057	5.056	4.541	4.537	4.537
PG-19	16.357	16.403	16.402	11.503	11.483	11.481	9.513	9.498	9.497
Newsroom	12.228	12.233	12.232	9.339	9.324	9.323	7.888	7.878	7.877
C4	14.112	14.128	14.128	10.854	10.842	10.841	9.321	9.311	9.311
LAMBADA	25.992	26.031	26.021	20.969	20.924	20.911	18.885	18.872	18.870
Average	12.302	12.317	12.315	9.223	9.208	9.206	8.000	7.991	7.990

E.3 Hyperparameter selection

Layer-pair grids of the other models. Figure 11 shows the layer-pair grids of Gemma 3 1B and 4B, Llama 3.1 8B, and Qwen3 8B, computed as in Figure 4(A, B).

Stability of the selection. To judge whether the remaining differences between the two landscapes matter for selection, we compare them with the differences between two runs of sequential recirculation at different batch sizes, on a subset of the pairs (Appendix C.5). Table 13 shows that Jacobi recirculation ranks the pairs as consistently with sequential recirculation as sequential recirculation does with itself. On the subsets, the Spearman correlation is 0.968 to 0.999 between Jacobi recirculation and sequential recirculation, and 0.965 to 0.999 between two runs of sequential recirculation. On the three Gemma models, the best pair of sequential recirculation leads the second-best one by 0.012 to 0.030 percentage points, above the median rounding scale, and Jacobi recirculation selects the same pair. On Llama 3.1 8B and Qwen3 8B, this lead is only 0.004 points, below the median rounding scale of 0.006 and 0.009 points. A second run of sequential recirculation with a different batch size could thus also select a different pair. Here Jacobi recirculation selects the pair that sequential recirculation ranks third or fourth.

Table 13: Stability of layer-pair selection on 128 arXiv training windows. Spearman correlations of the perplexity reduction over all pairs, and over the subset of pairs rerun with sequential recirculation at a smaller batch size. Lead is the difference in perplexity reduction between the best and second-best pair of sequential recirculation, and the rounding scale is the median difference between the two runs of sequential recirculation on the subset, both in percentage points.

	All pairs		Subset				
Model	Pairs	JR–SR	Pairs	SR–SR	JR–SR	Best SR / JR	Lead / rounding scale
Gemma 3 1B	300	0.997	25	0.998	0.993	$(24, 22)$ / $(24, 22)$	0.030 / 0.011
Gemma 3 4B	528	0.998	33	0.999	0.999	$(15, 8)$ / $(15, 8)$	0.018 / 0.009
Gemma 3 12B	1,081	0.995	23	0.991	0.995	$(26, 10)$ / $(26, 10)$	0.012 / 0.008
Llama 3.1 8B	465	0.993	30	0.985	0.983	$(12, 2)$ / $(14, 2)$	0.004 / 0.006
Qwen3 8B	595	0.977	38	0.965	0.968	$(18, 10)$ / $(18, 7)$	0.004 / 0.009

Coefficient sweeps. Table 14 lists the coefficient selected by each computation in the nine sweeps of Section 4.1. With two rounds, Jacobi recirculation selects the same coefficient as sequential recirculation in seven sweeps. The two exceptions are the sweeps in which the two best coefficients of sequential recirculation differ by at most 1.3×10^{-4} nats in mean NLL, comparable to the rounding scale of the layer-pair sweeps. With three rounds, Jacobi recirculation selects an adjacent coefficient in three sweeps.

Where the two computations differ. Figure 12 plots the difference between the two computations for every pair of the layer-pair sweeps against the effect of recirculation. Among the pairs that improve perplexity, Jacobi recirculation and sequential recirculation differ by at most 0.03% to 0.05%, close to the rounding scale. Larger differences occur only at pairs that make perplexity

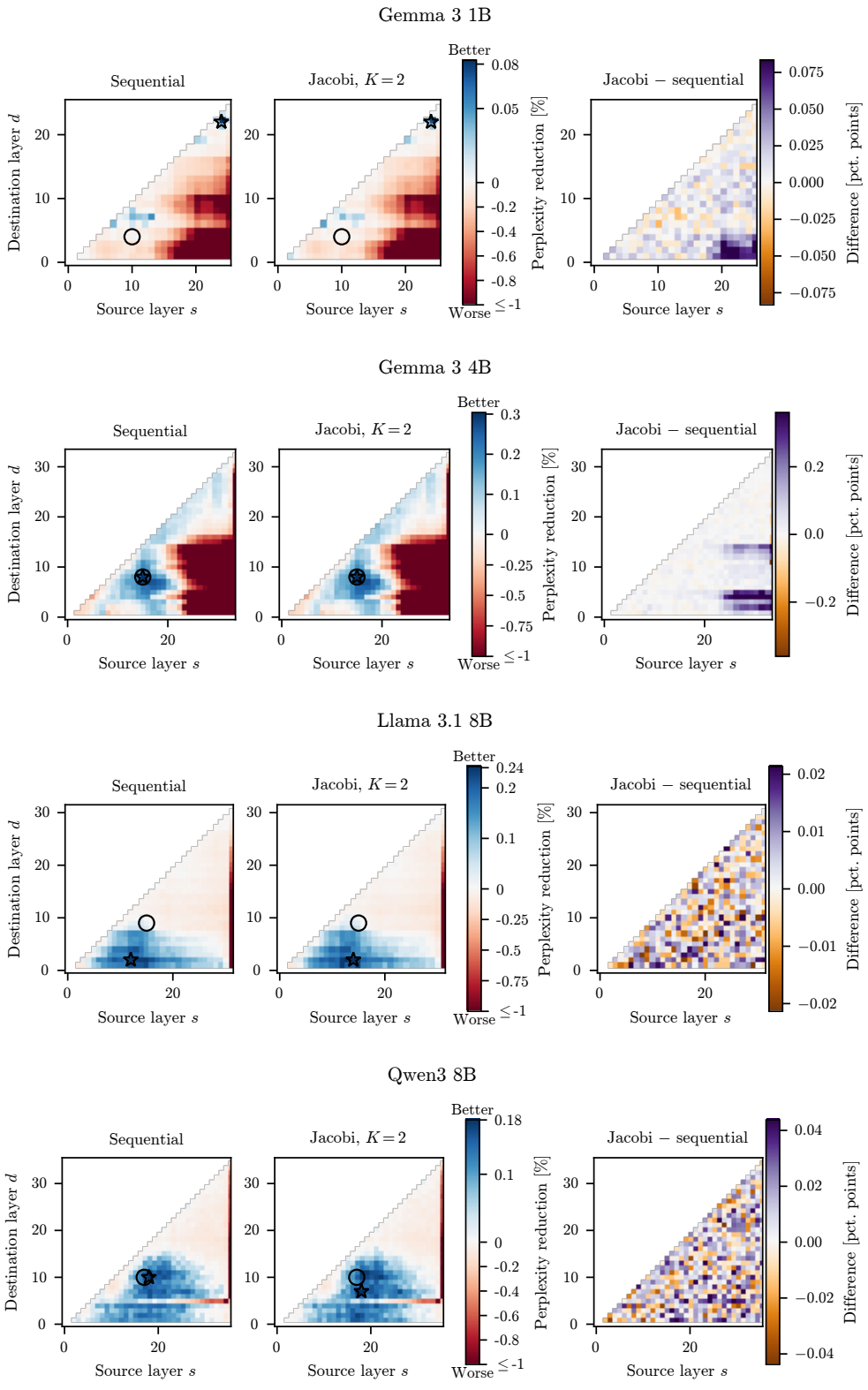


Figure 11: Training-free perplexity reduction on every layer pair with sequential recirculation and Jacobi recirculation with $K=2$, and their difference, for Gemma 3 1B and 4B, Llama 3.1 8B, and Qwen3 8B, on 128 arXiv training windows with the mixing coefficient and rule of the perplexity setting of each model (Table 6). Markers as in Figure 4(A, B), with the circle at the perplexity setting of each model.

Table 14: Coefficient selected by SR and by JR with K rounds in the coefficient sweeps, over $\alpha \in \{0.02, 0.03, 0.05, 0.07, 0.1, 0.15, 0.2, 0.3, 0.5\}$ on 128 arXiv training windows. Gap is the difference in mean NLL between the best and second-best coefficient of SR, in units of 10^{-4} nats.

Model	(s, d)	SR	JR K=2	JR K=3	Gap
Gemma 3 1B	$(10, 4)$	0.03	0.02	0.03	1.3
	$(24, 22)$	0.15	0.15	0.15	1.0
Gemma 3 4B	$(15, 8)$	0.15	0.15	0.15	4.6
Gemma 3 12B	$(22, 13)$	0.15	0.15	0.1	3.0
	$(26, 10)$	0.1	0.1	0.1	1.9
Llama 3.1 8B	$(15, 9)$	0.05	0.05	0.07	0.1
	$(12, 2)$	0.1	0.1	0.1	2.0
Qwen3 8B	$(17, 10)$	0.15	0.15	0.2	2.8
	$(18, 10)$	0.15	0.2	0.15	0.4

worse, up to 0.66% on 4B and 0.82% on 12B. Such pairs are not selected, so these differences do not affect hyperparameter selection.

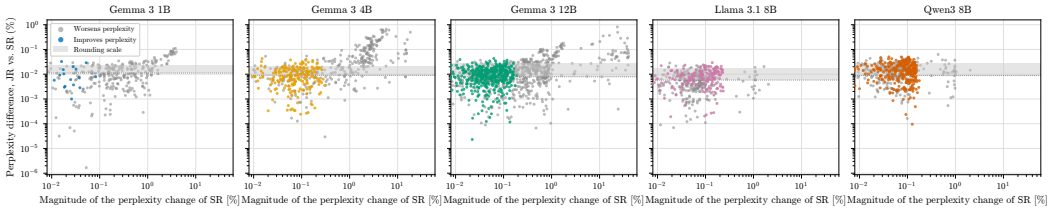


Figure 12: Relative difference in perplexity between Jacobi recirculation with K=2 and sequential recirculation for every layer pair of the sweeps in Section 4.1, versus the magnitude of the perplexity change that sequential recirculation causes. Colored points improve perplexity and gray points make it worse. The band shows the rounding scale, from the median to the 95th percentile of the difference between two runs of sequential recirculation with different batch sizes.

E.4 Mixer training

Gradient accuracy on individual windows. Figure 4(C, D) averages the cosine over eight training windows. On most windows the two-round gradient is close to the exact one, but a few windows fall well below the average, with two-round cosines as low as 0.67 on 1B and 0.78 on 4B in fp32. Such windows are not a separate class, since their cache error, contraction rate, and mixing coefficients are comparable to those of the other windows. When such a window has a large gradient, it can dominate the gradient of the batch. At one 1B checkpoint, a window whose two-round gradient is ten times larger than the exact one reduces the cosine of the batch gradient to 0.02, and a third round restores it to 0.996. On 4B, additional rounds do not always help at once. The cosine of one window drops from 0.94 after three rounds to 0.70 after four before reaching 1.00 after six, consistent with the Kq^{K-1} term of the bound in Section 3.3, which can grow before it decays. The mixers trained through two rounds nevertheless match those trained through sequential recirculation (Table 2), so these occasional inaccurate gradients do not affect the result of training.

E.5 Generation

Mixers trained through sequential recirculation. Table 3 evaluates the GSM8K mixers trained through Jacobi recirculation. For the Llama 3.1 8B and Qwen3 8B mixers trained through sequential recirculation, replacing sequential prefill with two-round Jacobi prefill also leaves pass@1 unchanged within seed variation, at 48.8 ± 0.8 versus 48.8 ± 1.4 on Llama 3.1 8B and 83.0 ± 0.6 versus 83.0 ± 0.4 on Qwen3 8B.

Propagation through autoregressive decoding. GSM8K and SVAMP continue from prompt prefill into free generation, and we analyze GSM8K. Table 15 separates task accuracy from trajectory

Table 15: Sensitivity of free generation to prompt-prefill differences on GSM8K. Each JR row compares sequential prefill with K-round Jacobi prefill, followed by the same greedy token-by-token decoder. SR/SR compares two sequential prefills under different batch splits. Pass@1 reports the two accuracies. Exact text and answer match are the percentages of questions with identical generated token sequences and extracted answers. The next column reports the percentage whose first divergence occurs within ten generated tokens, and the last column reports the median first-divergence position among questions that diverge. JR rows average three mixer seeds; SR/SR rows use seed 0, so their pass@1 values differ from the multi-seed averages.

Model	Comparison	Pass@1 SR / other	Exact text	Answer match	Diverged by 10	Median divergence
4B	JR $K{=}2$	40.3 / 38.5	22	58	34	12
	JR $K{=}3$	40.3 / 40.0	39	69	21	18
	JR $K{=}4$	40.3 / 40.1	49	74	16	21
	JR $K{=}6$	40.3 / 39.4	52	76	14	22
	SR/SR	42.2 / 42.1	75	89	7	25
12B	JR $K{=}2$	66.7 / 66.5	16	73	45	9
	JR $K{=}3$	66.7 / 66.7	36	79	27	13
	JR $K{=}4$	66.7 / 67.6	50	84	18	17
	JR $K{=}6$	66.7 / 67.0	57	87	14	22
	SR/SR	65.7 / 66.3	57	86	13	22
Llama 3.1 8B	JR $K{=}2$	47.5 / 47.8	42	72	22	16
	JR $K{=}3$	47.5 / 47.5	58	80	13	20
	SR/SR	48.4 / 48.3	87	94	3	32
Qwen3 8B	JR $K{=}2$	82.6 / 82.6	45	90	17	24
	JR $K{=}3$	82.6 / 82.6	45	89	17	25
	SR/SR	82.8 / 83.0	69	94	8	28

agreement. With two Jacobi rounds, complete generations match the sequential reference on 16%–45% of questions, while final answers match on 58%–90%. In most cases, additional rounds increase both forms of agreement and delay the first divergence. Greedy decoding is also sensitive to numerical variation: two mathematically equivalent sequential runs under different batch splits produce identical text on only 57%–87% of questions. Their final answers still agree on 86%–94%. Thus, different token trajectories need not imply a material change in task accuracy.

F Efficiency: supplementary results

This appendix derives the cost model of Section 4.2 for each model (Appendix F.1), lists the complete timing and memory measurements on the MI210 (Appendix F.2), and repeats the prefill timings on NVIDIA A100 and H100 GPUs (Appendix F.3).

F.1 Cost model

This subsection derives the analytical quantities of Section 4.2: the layer evaluations, the FLOPs and bytes of each computation, the roofline bounds and the ideal speed-up, the attained fractions of the hardware peaks, and the FLOPs ratios of the end-to-end scenarios.

Conventions. We count FLOPs and bytes analytically from the model geometry. A multiply and an add count as one FLOP each, so a matrix product costs 2 FLOPs per parameter per token, consistent with the hardware peak. Let N be the number of parameters of one transformer layer, d_{attn} the total query width, the number of query heads multiplied by the head dimension, and c the number of bytes of the key and value of one position in one layer. We drop low-order terms such as normalizations, the softmax, rotary embeddings, and activation functions, and count the LM head separately.

Layer evaluations. We count one unit for processing all T tokens through one layer. The plain forward pass evaluates each layer once, so $E_{\mathrm{plain}} = L$. Sequential recirculation evaluates layers $0, \ldots, d$ once in the batched prefix. Each layer above d processes every token twice, once in the source pass and once in the recirculated pass, so $E_{\mathrm{seq}} = (d+1) + 2(L-d-1)$. Jacobi recirculation first runs layers $0, \ldots, s$ to initialize the source states. Each intermediate round runs the recirculated

and the source pass through layers $d+1, \ldots, s$, and the final round runs both passes through layers $d+1, \ldots, L-1$. This gives $E_{\text{Jac}} = (s+1)+2(K-1)p+2(L-d-1)$. Layers $d+1, \ldots, s$ are thus evaluated $2K+1$ times, layers above s twice, and layers at or below d once, and each additional round costs $2p$ layer evaluations.

FLOPs. The matrix products of one layer evaluation cost $2NT$ FLOPs. Query t of layer ℓ attends to $|S_\ell(t)|$ keys, $t+1$ in a global layer and $\min(t+1, W)$ in a sliding-window layer, and its score and value products cost $4\,|S_\ell(t)|\,d_{\text{attn}}$ FLOPs. Summed over the sequence, a global layer attends to $A_{\text{g}} = T(T+1)/2$ keys and a sliding-window layer to $A_{\text{w}} = W(W+1)/2 + (T-W)W$ keys when $T > W$. A computation with the multiset $\mathcal{E}$ of layer evaluations therefore costs

$$F = 2NT\,|\mathcal{E}| + 4\,d_{\text{attn}} \textstyle\sum_{\ell \in \mathcal{E}} A_\ell,$$

with $|\mathcal{E}| = E_{\text{plain}}$, E_{seq}, or E_{Jac}. For sequential recirculation this holds because each step processes two tokens through the layers above d. For B sequences, all counts are multiplied by B. The LM head costs $2VH$ FLOPs for every position at which the logits are needed, where V is the vocabulary size.

Memory traffic. Weights are read at 2 bytes per bf16 parameter. The plain forward pass and Jacobi recirculation read the weights of a layer once per layer evaluation, $2N|\mathcal{E}|$ bytes in total. Sequential recirculation reads the weights of layers $0, \ldots, d$ once in the prefix, but those of every layer above d once per token step, since each step processes only two rows. Its weight traffic is therefore $2N\big[(d+1) + (L-d-1)T\big]$ bytes, which all B sequences of a step share. The traffic of keys and values follows the attended sets. In the plain pass and Jacobi recirculation, flash attention reads each block of keys and values once per block of 128 queries, which gives $c\,A_\ell/128$ bytes per layer evaluation. In sequential recirculation, each step reads the attended keys and values of every layer above d once for both rows, which gives $c\,A_\ell$ bytes per layer. Activations add 12 vectors of H values per token and layer evaluation, a small term. The score matrix is not materialized, so attention adds no further traffic.

Roofline bound and ideal speed-up. On a device with peak compute P and bandwidth β_{mem}, a computation with F FLOPs and Q bytes takes at least $\max(F/P,\, Q/\beta_{\text{mem}})$. We use the peaks of the MI210, $P = 181$ TFLOP/s in bf16 and $\beta_{\text{mem}} = 1.6$ TB/s. The ideal speed-up is the ratio of the bounds of the two recirculations,

$$S_{\text{ideal}} = \frac{\max(F_{\text{seq}}/P,\, Q_{\text{seq}}/\beta_{\text{mem}})}{\max(F_{\text{Jac}}/P,\, Q_{\text{Jac}}/\beta_{\text{mem}})}.$$

At $T = 1{,}024$ and $B = 1$, sequential recirculation is memory-bound and Jacobi recirculation compute-bound on all five models (Table 17), so $S_{\text{ideal}} = (P/\beta_{\text{mem}})(Q_{\text{seq}}/F_{\text{Jac}})$. The weight traffic of the layers above d dominates Q_{seq}, and the matrix products dominate F_{Jac}. Keeping only these terms gives

$$S_{\text{ideal}} \approx \frac{P}{\beta_{\text{mem}}} \cdot \frac{2N(L-d-1)T}{2NT\,B\,E_{\text{Jac}}} = \frac{P}{\beta_{\text{mem}}} \cdot \frac{L-d-1}{B\,E_{\text{Jac}}}.$$

The sequence length cancels because both terms grow linearly with T. The batch appears only in the denominator, because the sequences of a step share the weight reads of sequential recirculation but each add arithmetic to Jacobi recirculation. As B grows, sequential recirculation also becomes compute-bound, and the ideal speed-up approaches the FLOPs ratio $F_{\text{seq}}/F_{\text{Jac}} \approx E_{\text{seq}}/E_{\text{Jac}} < 1$.

Attained fractions of the peaks. For a measured time t, the model FLOP utilization is $\text{MFU} = F/(Pt)$ and the memory bandwidth utilization is $\text{MBU} = Q/(\beta_{\text{mem}}t)$. When sequential recirculation is memory-bound and Jacobi recirculation compute-bound, the measured speed-up is

$$\frac{t_{\text{seq}}}{t_{\text{Jac}}} = \frac{Q_{\text{seq}}/(\beta_{\text{mem}}\,\text{MBU}_{\text{seq}})}{F_{\text{Jac}}/(P\,\text{MFU}_{\text{Jac}})} = S_{\text{ideal}} \cdot \frac{\text{MFU}_{\text{Jac}}}{\text{MBU}_{\text{seq}}}.$$

It exceeds S_{ideal} whenever Jacobi recirculation attains a larger fraction of its peak than sequential recirculation does of its own.

End-to-end scenarios. For the FLOPs ratios of Table 4, we sum these counts over the actual inputs of each task, with the layer pair of the task. The LM head is counted at every position for perplexity, and only at the readout position for the instruction task and the GSM8K prompts. On GSM8K, each decoded token adds the same cost to both computations, namely $L + (L-d-1)$ layer evaluations of one token, attention over the growing context in these layers, and the LM head. This shared decoding cost brings the FLOPs ratio of GSM8K close to one.

Training. For a training step, we count the backward pass through the frozen layers at the cost of the forward pass. This holds for the linear layers, where only input gradients are needed, and

understates the attention backward, which also forms the gradients of the queries, keys, and values and recomputes the softmax. The forward and backward pass of the LM head are added. Under this estimate, the ratio of Jacobi to sequential FLOPs per step is 1.31–1.32 for the arXiv mixers, 1.48 and 1.68 for the GSM8K mixers on Gemma 3 4B and 12B, and 1.40 and 1.35 for the GSM8K mixers on Llama 3.1 8B and Qwen3 8B. We expect the omitted attention-backward costs to have a small effect on these ratios at the sequence lengths considered.

Worked example. We illustrate the counts on Gemma 3 12B at $T = 1{,}024$, $B = 1$, and $K = 2$ with the pair $(22, 13)$, and list the same quantities for all five models in Tables 16 and 17. From the geometry in Table 5, one layer has $N = H\,d_{\text{attn}} + 2H\,n_{\text{kv}}d_{\text{head}} + d_{\text{attn}}H + 3HI = 224.1\text{M}$ parameters, where 16 query heads of dimension $d_{\text{head}} = 256$ give $d_{\text{attn}} = 4{,}096$, there are $n_{\text{kv}} = 8$ key-value heads, and $I = 15{,}360$ is the width of the feed-forward layer. The key and value of one position take $c = 2 \cdot 2\,n_{\text{kv}}d_{\text{head}} = 8{,}192$ bytes. Since the sliding window $W = 1{,}024$ covers the whole sequence, every layer attends to $A = T(T+1)/2 = 524{,}800$ keys. The pair gives $E_{\text{seq}} = 14 + 2 \cdot 34 = 82$ and $E_{\text{Jac}} = 23 + 2 \cdot 9 + 2 \cdot 34 = 109$. Jacobi recirculation then costs $2NT \cdot 109 = 50.0$ TFLOPs in its matrix products and $4 \cdot 4{,}096 \cdot 524{,}800 \cdot 109 = 0.94$ TFLOPs in attention, so $F_{\text{Jac}}/P = 282$ ms. It moves 48.9 GB of weights, 3.7 GB of keys and values, and 10.3 GB of activations, so $Q_{\text{Jac}}/\beta_{\text{mem}} = 39$ ms, and it is compute-bound. Sequential recirculation costs 38.3 TFLOPs, or 212 ms at peak compute. Its weight traffic is $2N(14 + 34 \cdot 1{,}024) = 15{,}613$ GB and its key-value traffic 147 GB, so $Q_{\text{seq}}/\beta_{\text{mem}} = 9{,}853$ ms, and it is memory-bound. The ideal speed-up is $9{,}853/281.6 = 35.0$, and the approximation above gives $(P/\beta_{\text{mem}})(L - d - 1)/E_{\text{Jac}} = 113.1 \cdot 34/109 = 35.3$. On all five models, the approximation lies within 3% of the full count.

Batch size. With B sequences, the weight traffic stays the same for every computation, while the FLOPs and the traffic of keys, values, and activations grow B-fold. The ideal speed-up therefore falls below one at $B = 51$, 58, and 53 for Gemma 3 1B, 4B, and 12B, and at $B = 42$ and 43 for Llama 3.1 8B and Qwen3 8B. For large B, it approaches the FLOPs ratio $F_{\text{seq}}/F_{\text{Jac}}$, which is 0.72 to 0.75.

Table 16: Quantities of the cost model at T=1,024 and K=2. N is the number of parameters per layer, c the bytes of the key and value of one position in one layer, and A_{g} and A_{w} the keys attended per layer in a global and a sliding-window layer.

Model	$(s, d), p$	N	c (bytes)	Global / sliding layers	A_{g} / A_{w}	E_{plain} / E_{seq} / E_{Jac}
Gemma 3 1B	$(10, 4), 6$	26.8M	1,024	4 / 22	524,800 / 393,472	26 / 47 / 65
Gemma 3 4B	$(15, 8), 7$	94.4M	4,096	5 / 29	524,800 / 524,800	34 / 59 / 80
Gemma 3 12B	$(22, 13), 9$	224.1M	8,192	8 / 40	524,800 / 524,800	48 / 82 / 109
Llama 3.1 8B	$(15, 9), 6$	218.1M	4,096	32 / 0	524,800 / —	32 / 54 / 72
Qwen3 8B	$(17, 10), 7$	192.9M	4,096	36 / 0	524,800 / —	36 / 61 / 82

Table 17: Cost model at T=1,024, B=1, and K=2, excluding the LM head, on an AMD MI210 (P=181 TFLOP/s, β_{mem}=1.6 TB/s). FLOPs are split into matrix products and attention, and bytes into weights, keys and values, and activations. The larger of F/P and Q/β_{mem}, in bold, is the roofline bound. S_{ideal} is the ratio of the bounds of sequential and Jacobi recirculation.

Model	Computation	FLOPs (T): matmul / attn.	Bytes (GB): weights / KV / act.	F/P (ms)	Q/β_{mem} (ms)	S_{ideal}
Gemma 3 1B	plain	1.429 / 0.044	1.40 / 0.09 / 0.74	**8.1**	1.4	
	Jacobi	3.573 / 0.111	3.49 / 0.22 / 1.85	**20.3**	3.5	
	sequential	2.583 / 0.080	1,154 / 9.0 / 0.74	14.7	**727.6**	35.8×
Gemma 3 4B	plain	6.571 / 0.146	6.42 / 0.57 / 2.14	**37.1**	5.7	
	Jacobi	15.462 / 0.344	15.10 / 1.34 / 5.05	**87.3**	13.4	
	sequential	11.403 / 0.254	4,834 / 53.9 / 2.14	64.4	**3,056.0**	35.0×
Gemma 3 12B	plain	22.033 / 0.413	21.52 / 1.61 / 4.53	**124.0**	17.3	
	Jacobi	50.034 / 0.937	48.86 / 3.66 / 10.32	**281.6**	39.3	
	sequential	37.640 / 0.705	15,613 / 146.6 / 4.53	211.9	**9,852.7**	35.0×
Llama 3.1 8B	plain	14.294 / 0.275	13.96 / 0.54 / 3.22	**80.5**	11.1	
	Jacobi	32.161 / 0.619	31.41 / 1.21 / 7.28	**181.1**	24.9	
	sequential	24.121 / 0.464	9,831 / 47.5 / 3.22	135.8	**6,176.2**	34.1×
Qwen3 8B	plain	14.225 / 0.310	13.89 / 0.60 / 3.62	**80.3**	11.3	
	Jacobi	32.401 / 0.705	31.64 / 1.38 / 8.29	**182.9**	25.8	
	sequential	24.103 / 0.524	9,883 / 53.9 / 3.62	136.1	**6,212.6**	34.0×

Table 18: Compile and CUDA-graph ablation of the plain forward pass and sequential recirculation (ms; 1B and 12B, T=1,024 and 16k, B=1). The setting used in all measurements is the first row. These measurements are separate from those of Table 21. With compilation but without graph capture, the plain forward pass recompiles at every call, so its time is not shown.

	1B, 1k		1B, 16k		12B, 1k		12B, 16k	
compile / graph	plain	seq.	plain	seq.	plain	seq.	plain	seq.
on / on	22	4,484	275	74,338	223	23,575	3,390	389,051
off / on	35	14,211	376	229,239	272	41,050	4,088	695,753
off / off	47	23,630	379	373,305	277	47,244	—	—
on / off	—	5,410	—	89,236	—	24,444	—	—

Table 19: Prefill cost for Gemma 3, Llama 3.1 8B, and Qwen3 8B on one MI210 at $T = 1{,}024$, $B = 1$, and $K = 2$, excluding the LM head. Triples list plain / Jacobi / sequential. FLOPs and roofline bounds are computed analytically (Appendix F.1). Speed-up is the measured sequential/Jacobi runtime ratio, with S_{ideal} in parentheses.

Model	FLOPs (TFLOP)	Runtime	Roofline bound	Speed-up	MFU (Jacobi)	MBU (seq.)
1B	1.47 / 3.68 / 2.66	22.1 ms / 53.7 ms / 4.48 s	8.1 ms / 20.3 ms / 0.73 s	83× (36×)	0.38	0.16
4B	6.72 / 15.8 / 11.7	74.0 ms / 170 ms / 9.58 s	37.1 ms / 87.3 ms / 3.06 s	56× (35×)	0.51	0.32
12B	22.4 / 51.0 / 38.3	219 ms / 490 ms / 23.1 s	124 ms / 282 ms / 9.85 s	47× (35×)	0.57	0.43
Llama 3.1 8B	14.6 / 32.8 / 24.6	138 ms / 303 ms / 14.5 s	80.5 ms / 181 ms / 6.18 s	48× (34×)	0.60	0.43
Qwen3 8B	14.5 / 33.1 / 24.6	141 ms / 315 ms / 15.2 s	80.3 ms / 183 ms / 6.21 s	48× (34×)	0.58	0.41

Table 20: Kernel profiling on one MI210 in bf16 at $T = 1{,}024$, $B = 1$, and $K = 2$, from a profiled run whose wall time differs slightly from Table 19. Sequential columns report the wall time, the time in GEMM kernels including the attention products, the weights read, and the achieved bandwidth per paired token step. The Jacobi column reports the fraction of the run time spent in GEMMs. The M=2 MBU and M=1024 MFU columns are isolated matrix-product benchmarks with matching shapes, normalized by peak bandwidth and peak compute, respectively; M is the number of input rows.

	Sequential step					Jacobi recirculation	
Model	Step (ms)	GEMM (ms)	Weights (GB)	Achieved GB/s	M=2 MBU	GEMM share	M=1024 MFU
1B	4.2	3.9	1.13	269	0.23	71%	0.54
4B	9.2	8.3	4.72	515	0.40	83%	0.59
12B	22.3	21.1	15.2	683	0.51	89%	0.63

F.2 Measurements

Each path runs in its fastest configuration. Table 18 times the plain and sequential recirculation with torch.compile and CUDA-graph capture switched on and off (B=1, median of three repetitions, or the slower of two at 16k); the configuration used throughout, compile and graph on, is the fastest for both computations at both lengths (for the plain forward pass, compilation without graph capture recompiles at every call and is excluded). For Jacobi recirculation the same holds in a separate measurement: at T=8k the compiled and captured 1B path takes 346 ms against 518 ms uncompiled, and at 16k 689 ms against 981 ms. Apart from the larger cache buckets of 12B at 8k tokens (Appendix C.2), no configuration of the prefill timings is switched between lengths or batch sizes.

Prefill measurements. Table 19 lists the prefill measurements behind Section 4.2: the FLOPs and roofline bounds computed with the counting rules above, the measured runtimes and speed-ups, and the hardware utilization of both computations.

Kernel profiling. Table 20 profiles the prefill of both computations. In isolation, the matrix products of sequential recirculation reach only 23–51% of peak bandwidth with two input rows, whereas those of Jacobi recirculation reach 54–63% of peak compute with 1,024 rows.

Timing grid. Table 21 lists the measurements behind Figure 5 and Table 19, excluding and including the LM head. Peak memory is reported in Table 30. Llama 3.1 8B and Qwen3 8B, measured with the same implementation and protocol, follow the Gemma 3 curves closely: their speed-ups stay between $42\times$ and $50\times$ up to 16k tokens and fall to $2.6\times$ and $2.7\times$ at $B{=}32$.

Long contexts. Table 22 lists the measurements behind the long-context results of Section 4.2. For each model and sequence length, we find the largest batch that fits sequential recirculation on one device by halving the batch from above, with each computation in its own process. Jacobi recirculation fits the same batch in every case, so both are timed at this batch. Both computations are compiled and captured. Figure 13 shows the speed-up over the batch at each sequence length, together with S_{ideal}. The measured speed-up exceeds the ideal one everywhere and stays above one where the ideal one falls below it.

Layer pairs. Figure 14 shows the measured speed-up of every layer pair of the sweeps in Section 4.1, at the batch of the sweeps, against the factor $(L-d-1)/E_{\text{Jac}}$ through which the layer pair and the number of rounds enter S_{ideal}. The logarithms of the two are correlated with coefficients of 0.94 to 0.99, so the layer pair affects the speed-up as the cost model predicts.

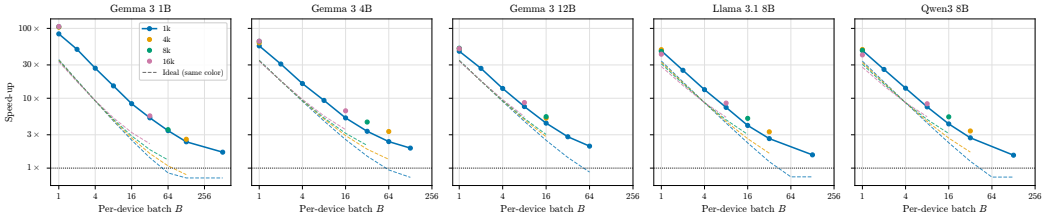


Figure 13: Prefill speed-up of Jacobi over sequential recirculation versus the per-device batch on one MI210 in bf16 with $K{=}2$, excluding the LM head. Markers show measurements at each sequence length, at $B{=}1$ and at the largest batch that fits sequential recirculation, and at 1k tokens at every batch in between (solid line). Dashed lines of the same color show S_{ideal}. The dotted line marks a speed-up of one.

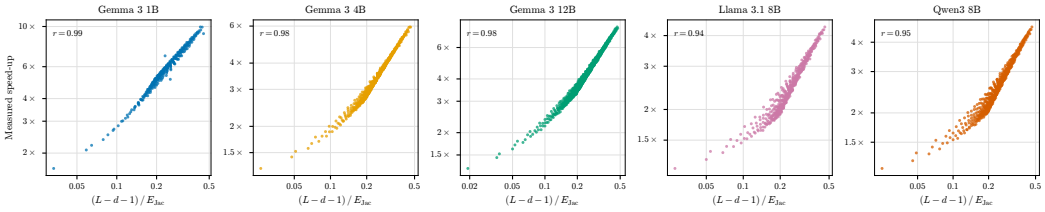


Figure 14: Measured speed-up of Jacobi recirculation with $K{=}2$ over sequential recirculation for every layer pair of the sweeps in Section 4.1, versus the cost-model factor $(L-d-1)/E_{\text{Jac}}$, with the Pearson correlation of the logarithms.

Data parallelism. Table 23 processes the same 32 windows of 1,024 tokens either on eight devices with four windows each or on one device as a batch of 32. Each of the eight devices runs the single-device computation at $B{=}4$, so the speed-up matches or exceeds the single-device value at $B{=}4$. On one device, the batch of 32 reduces it to $5.3\times$, $3.4\times$, and $2.8\times$. Data parallelism therefore preserves the speed-up of the per-device batch.

End-to-end evaluation at a common batch. Table 24 repeats the instruction-task evaluations of Table 4 with 32 instead of 64 sequences per forward pass, the batch of the perplexity evaluations. The instruction task obtains speed-ups of $2.1\times$ and $1.5\times$ on 4B-IT and 12B-IT. GSM8K is not repeated, since its prompts are grouped by length and rarely fill a batch of 32.

Table 21: Measured prefill times on one AMD MI210 in bf16 with K=2, over sequence length at B=1 and over the per-device batch at T=1,024. Times in seconds excluding the LM head (median of five runs, or the slower of two for sequential recirculation beyond 4k tokens); speed-ups of Jacobi recirculation over sequential recirculation excluding and including the LM head. With the LM head, the 1B run at B=128, the 4B run at B=64, and the 12B run at B=32 do not fit in memory.

			Time (s)			Speed-up	
Model	T	B	Plain	Jacobi	Sequential	w/o head	with head
1B	1k	1	0.022	0.054	4.48	83.4×	65.5×
	2k	1	0.037	0.090	9.14	101.8×	76.5×
	4k	1	0.071	0.174	18.1	104.2×	77.7×
	8k	1	0.139	0.343	36.3	105.8×	78.8×
	16k	1	0.277	0.698	74.3	106.5×	79.4×
	1k	2	0.036	0.088	4.41	50.1×	38.1×
	1k	4	0.068	0.167	4.50	26.9×	20.1×
	1k	8	0.131	0.321	4.82	15.0×	11.0×
	1k	16	0.249	0.609	5.09	8.4×	6.2×
	1k	32	0.493	1.21	6.36	5.2×	3.9×
	1k	64	0.994	2.45	8.42	3.4×	2.7×
	1k	128	1.99	4.99	11.9	2.4×	—
4B	1k	1	0.074	0.170	9.58	56.2×	48.6×
	2k	1	0.139	0.321	20.1	62.7×	54.0×
	4k	1	0.281	0.652	40.5	62.1×	53.2×
	8k	1	0.531	1.24	81.2	65.4×	55.7×
	16k	1	1.06	2.50	163.6	65.6×	56.3×
	1k	2	0.138	0.319	9.87	30.9×	26.3×
	1k	4	0.278	0.643	10.5	16.2×	14.3×
	1k	8	0.501	1.16	10.8	9.3×	8.1×
	1k	16	1.01	2.35	12.3	5.2×	4.7×
	1k	32	2.02	4.69	15.8	3.4×	3.1×
	1k	64	3.96	9.19	22.0	2.4×	—
12B	1k	1	0.219	0.490	23.1	47.1×	44.5×
	2k	1	0.405	0.907	47.5	52.4×	48.8×
	4k	1	0.825	1.87	95.7	51.2×	47.6×
	8k	1	1.64	3.70	192.5	52.0×	49.1×
	16k	1	3.35	7.62	391.5	51.4×	48.7×
	1k	2	0.401	0.898	24.1	26.8×	25.0×
	1k	4	0.811	1.83	25.3	13.8×	13.0×
	1k	8	1.60	3.64	27.7	7.6×	7.2×
	1k	16	3.12	7.03	31.0	4.4×	4.2×
	1k	32	6.44	14.6	40.9	2.8×	—
Llama 3.1 8B	1k	1	0.138	0.303	14.5	47.8×	45.0×
	2k	1	0.272	0.599	29.1	48.5×	45.6×
	4k	1	0.545	1.21	59.8	49.6×	46.8×
	8k	1	1.17	2.61	122.4	46.9×	45.2×
	16k	1	2.70	6.02	257.2	42.7×	41.0×
	1k	2	0.265	0.584	14.7	25.1×	23.7×
	1k	4	0.522	1.16	15.4	13.3×	12.6×
	1k	8	1.00	2.22	16.4	7.4×	7.1×
	1k	16	2.05	4.56	18.6	4.1×	3.9×
	1k	32	3.99	8.92	23.6	2.6×	2.6×
Qwen3 8B	1k	1	0.141	0.315	15.2	48.4×	45.7×
	2k	1	0.274	0.613	30.4	49.6×	46.9×
	4k	1	0.558	1.26	62.7	49.8×	47.0×
	8k	1	1.17	2.66	129.3	48.5×	45.7×
	16k	1	2.78	6.48	270.4	41.7×	40.9×
	1k	2	0.269	0.605	15.6	25.9×	24.3×
	1k	4	0.522	1.17	16.3	13.9×	13.2×
	1k	8	1.02	2.31	17.4	7.5×	7.2×
	1k	16	2.02	4.58	19.6	4.3×	4.1×
	1k	32	4.11	9.23	25.1	2.7×	2.6×

Table 22: Prefill at the largest batch B^* that fits sequential recirculation on one MI210, in bf16 with $K=2$, excluding the LM head. Times are seconds per sequence, and the speed-up is their ratio.

Model	T	B^*	Sequential	Jacobi	Speed-up
Gemma 3 1B	1k	512	0.0645	0.0381	1.7×
	4k	128	0.405	0.157	2.6×
	8k	64	1.16	0.327	3.6×
	16k	32	3.79	0.678	5.6×
Gemma 3 4B	1k	128	0.280	0.145	1.9×
	4k	64	1.98	0.591	3.3×
	8k	32	5.46	1.20	4.6×
	16k	16	16.6	2.51	6.6×
Gemma 3 12B	1k	64	0.916	0.444	2.1×
	4k	16	9.49	1.83	5.2×
	8k	16	20.4	3.74	5.4×
	16k	8	67.8	7.82	8.7×
Llama 3.1 8B	1k	128	0.438	0.283	1.5×
	4k	32	3.92	1.19	3.3×
	8k	16	13.4	2.61	5.2×
	16k	8	50.4	5.91	8.5×
Qwen3 8B	1k	128	0.445	0.290	1.5×
	4k	32	4.16	1.22	3.4×
	8k	16	14.5	2.67	5.4×
	16k	8	53.0	6.36	8.3×

Table 23: Eight-way data parallelism at $T=1{,}024$. Time in seconds for the plain forward pass, Jacobi recirculation, and sequential recirculation to process 32 windows as four per device on eight devices (maximum over devices) and as one batch of 32 on a single device, with the resulting speed-ups, against the single-device speed-up at $B=4$ of Table 21. The single-device batch of 32 is measured separately from Table 21.

	8 devices, $B=4$ each		1 device, $B=32$		1 device, $B=4$
Model	Plain / Jacobi / Sequential	Speed-up	Plain / Jacobi / Sequential	Speed-up	Speed-up
1B	0.068 / 0.165 / 4.79	29.0×	0.49 / 1.21 / 6.4	5.3×	26.9×
4B	0.274 / 0.633 / 10.59	16.7×	1.98 / 4.62 / 15.7	3.4×	16.2×
12B	0.812 / 1.822 / 25.40	13.9×	6.35 / 14.28 / 39.5	2.8×	13.8×

Table 24: End-to-end running time of complete evaluations at a common per-device batch, with SR and with JR ($K=2$) on MI210 devices in bf16. The instruction task (800 prompts), in a single process. The last column is the ratio of the FLOPs of the two computations.

Scenario	Model	Batch	SR	JR	Speed-up	FLOPs JR/SR
Instruction task	4B-IT	32	28 s	13 s	2.1×	1.51
	12B-IT	32	66 s	45 s	1.5×	1.49

Complete evaluations on other architectures. Table 25 lists the running times of Llama 3.1 8B and Qwen3 8B, which follow the same pattern as Gemma 3 in Table 4. Its instruction-task rows use the settings with $\alpha = 0.5$ of Table 11.

Prompt and decoding phases on GSM8K. Table 26 splits the GSM8K runtime into the prompt phase and the token-by-token decoding, measured in one session on the same devices. The prompt phase starts after tokenization and the allocation of the cache and includes the forward computation over the prompt and the selection of the first generated token. Decoding continues until the longest answer in a batch ends or reaches 320 tokens. The two recirculation methods decode nearly the same number of tokens and take the same time per step. The model without recirculation decodes up to 42% more steps, because without the trained mixer it often continues past the answer until the length

Table 25: End-to-end running time for Llama 3.1 8B and Qwen3 8B, measured and reported as in Table 4. Perplexity covers the arXiv test set.

Scenario	Model	Batch	Base	SR	JR	Speed-up	FLOPs JR/SR
Perplexity, arXiv	Llama 3.1 8B	32	1.8 h	9.7 h	3.7 h	2.6×	1.32
	Qwen3 8B	32	2.0 h	11.0 h	4.1 h	2.7×	1.33
Instruction task	Llama 3.1 8B Instruct	64	13 s	53 s	29 s	1.8×	1.27
	Qwen3 8B	64	10 s	46 s	28 s	1.6×	1.54
GSM8K	Llama 3.1 8B	64	—	1,146 s	911 s	1.3×	1.16
	Qwen3 8B	64	—	1,243 s	1,073 s	1.2×	1.13

limit. With one prompt at a time, measured on 16 problems, the prompt phase of Jacobi recirculation takes 0.06 and 0.13 s on 4B and 12B, against 0.86 and 1.68 s for sequential recirculation.

Table 26: GSM8K on MI210 devices in bf16, 1,319 prompts grouped by length into batches of up to 64, with the trained GSM8K mixers of Appendix C.6. Prompt phase and total time are summed over all batches. Decoding steps are summed over batches, each running until its longest answer ends or reaches 320 tokens.

Model	Computation	Prompt phase (s)	Decoding steps	Time per step (ms)	Mean generated tokens	Total (s)
Gemma 3 1B	Base	2.4	36,160	28.2	313	1,024
	Sequential	52.3	34,984	31.3	199	1,146
	Jacobi, K=2	6.1	35,006	30.0	194	1,055
Gemma 3 4B	Base	7.7	34,569	37.0	229	1,288
	Sequential	129	28,207	39.9	127	1,255
	Jacobi, K=2	22.1	27,593	39.7	128	1,117
Gemma 3 12B	Base	21.5	35,224	61.4	298	2,186
	Sequential	255	26,960	64.6	129	1,996
	Jacobi, K=2	62.2	25,934	64.6	126	1,737
Llama 3.1 8B	Base	13.6	32,363	32.8	187	1,074
	Sequential	275	24,393	35.7	104	1,146
	Jacobi, K=2	35.8	24,478	35.8	104	911
Qwen3 8B	Base	13.6	35,779	37.8	319	1,364
	Sequential	212	25,271	40.8	127	1,243
	Jacobi, K=2	34.8	25,332	41.0	127	1,073

Training time in detail. Table 27 compares the exact trainer with the Jacobi trainer without compilation, with `torch.compile` in its default mode, and in its reduce-overhead mode, which adds CUDA graphs. For the Jacobi trainer, the parentheses give the median time per step from the second step on and the time of the first step, which contains compilation and graph capture. The two compiled modes are indistinguishable, and Table 4 uses the default mode. The single-device runs process the whole batch of a step on one device, four problems for GSM8K and 32 windows for the arXiv mixers.

Cost of additional rounds. Table 28 times Jacobi recirculation with one to sixteen rounds at T=1,024 and B=1 for the three Gemma 3 models and for Llama 3.1 8B and Qwen3 8B (Figure 5, third panel). Every round beyond the first adds two passes over the layers between d and s, about 0.4 plain forward passes for these layer pairs, and the measured and ideal speed-ups decrease accordingly.

Memory. Table 30 reports the peak memory of prefill with 32 sequences of 1,024 tokens per device. All three computations hold the same result at the end, the complete decoding cache, with every position at global-attention layers and the last $W+1$ positions at sliding-window layers. Beyond the weights and this cache, both recirculation methods keep the destination and source states of every position, and Jacobi recirculation additionally holds the intermediate activations of all BT tokens at the layer it is computing. Without compilation, the two recirculation methods therefore need the same memory within 2% on all five models, 3% to 10% more than the plain forward pass. In training, the activation memory without recomputation is proportional to the layer evaluations E_{seq}

Table 27: Training time with the exact trainer and with the Jacobi trainer in three compilation modes, on AMD MI210 in bf16. Times are sums of the measured steps from the second to the last step. The parentheses give the median time per step and the time of the first step, both in seconds. In all three modes, the Jacobi trainer processes one window per device at a time, whereas it processes four in Table 4, two for 12B. The single-device perplexity rows of 1B and 4B instead use the largest batch that fits each trainer, 16 and 8 windows at a time for the exact trainer and 8 and 4 for the Jacobi trainer, in the compiled mode only. — marks settings not measured due to limited time.

Mixers	Model	Devices	Exact	Jacobi, eager	Jacobi, compiled	Jacobi, compiled + CUDA graphs
arXiv	1B	8	26.4 min (16; 22)	2.2 min (1.33; 3)	1.4 min (0.825; 60)	1.4 min (0.820; 59)
arXiv	1B	1	1.13 h (41.2; 47)	—	5.8 min (3.49; 59)	—
arXiv	4B	8	1.07 h (38.7; 45)	5.1 min (3.11; 4)	3.5 min (2.13; 68)	3.6 min (2.14; 67)
arXiv	4B	1	5.76 h (209; 215)	—	19.7 min (12.0; 77)	—
arXiv	12B	8	3.84 h (140; 146)	11.8 min (7.09; 9)	8.7 min (5.25; 95)	8.7 min (5.24; 93)
arXiv	12B	1	—	1.43 h (52; 53)	1.01 h (36.9; 122)	1.01 h (36.8; 121)
GSM8K	4B	4	1.89 h (8.5; 15)	7.9 min (0.584; 2)	6.6 min (0.461; 68)	6.3 min (0.458; 64)
GSM8K	4B	1	2.22 h (10; 16)	10.7 min (0.804; 2)	6.8 min (0.503; 65)	6.8 min (0.501; 64)
GSM8K	12B	4	3.36 h (15.1; 22)	16.3 min (1.20; 3)	13.1 min (0.937; 91)	13.3 min (0.984; 87)
GSM8K	12B	1	4.09 h (18.4; 24)	25.5 min (1.91; 3)	18.2 min (1.37; 90)	18.8 min (1.39; 86)
GSM8K	Llama 3.1 8B	4	2.75 h (12.4; 25)	—	11.7 min (0.873; 92)	—
GSM8K	Qwen3 8B	4	2.93 h (13.2; 27)	—	12.0 min (0.882; 120)	—

Table 28: Cost of the number of rounds at T=1,024 and B=1 on one AMD MI210 in bf16. Time of Jacobi recirculation excluding the LM head (median of five runs), the same time in units of the plain forward pass, and the measured and ideal speed-ups over sequential recirculation.

Model		K=1	K=2	K=3	K=4	K=6	K=8	K=16
1B	Jacobi time (ms)	44.5	53.7	62.0	71.4	91.6	109	185
	Plain forward passes	2.01	2.43	2.81	3.23	4.14	4.95	8.37
	Speed-up, measured	101	83	72	63	49	41	24
	Speed-up, ideal	44	36	30	26	21	17	10
4B	Jacobi time (ms)	141	170	199	222	281	340	568
	Plain forward passes	1.91	2.30	2.69	3.00	3.80	4.59	7.68
	Speed-up, measured	68	56	48	43	34	28	17
	Speed-up, ideal	42	35	30	26	21	17	10
12B	Jacobi time (ms)	412	490	576	646	791	969	1596
	Plain forward passes	1.88	2.23	2.62	2.95	3.61	4.42	7.28
	Speed-up, measured	56	47	40	36	29	24	14
	Speed-up, ideal	42	35	30	26	21	18	11
Llama 3.1 8B	Jacobi time (ms)	260	303	353	399	501	590	975
	Plain forward passes	1.88	2.19	2.56	2.89	3.63	4.28	7.06
	Speed-up, measured	56	48	41	36	29	25	15
	Speed-up, ideal	41	34	29	26	20	17	10
Qwen3 8B	Jacobi time (ms)	261	315	368	416	511	619	1037
	Plain forward passes	1.85	2.23	2.61	2.95	3.62	4.39	7.35
	Speed-up, measured	58	48	41	37	30	25	15
	Speed-up, ideal	41	34	29	25	20	17	10

and E_{Jac} of Section 4.2. Table 29 reports the peak memory of mixer training. With the same number of windows at a time, Jacobi training with $K = 2$ needs $1.07\times$ the memory of the exact trainer on 1B, $1.24\times$ on 4B, and $1.26\times$ on 12B. Each additional round adds 15% to 20% of the peak of the exact trainer, in line with the $2p$ layer evaluations that each round adds. On 12B, a third round exceeds the device memory with two windows at a time.

F.3 Other GPUs

Setup. We repeat the prefill measurements on one NVIDIA A100 80GB PCIe and one NVIDIA H100 80GB PCIe GPU, with the code, inputs, and protocol of Appendix C.7. The cost model uses the bf16 peaks $P = 312$ and 756 TFLOP/s and the bandwidths $\beta_{\mathrm{mem}} = 1.935$ and 2.0 TB/s. On these GPUs, the token steps of sequential recirculation can read the cache either with the attention used on the MI210 or with the fused decoding kernel of the FlashAttention library (Dao et al., 2022).

Table 29: Peak memory of mixer training in GB on one MI210, with 32 windows of 1,024 tokens per step and including the weights, for the exact sequential trainer and for Jacobi training with K rounds. Windows at a time is the number processed in one pass; OOM marks configurations that exceed the 64 GB of the device.

Model	Windows at a time	Sequential	Jacobi, $K{=}1$	Jacobi, $K{=}2$	Jacobi, $K{=}3$
1B	4	16.8	15.0	18.0	20.9
1B	8	30.9	27.5	33.4	39.4
1B	16	59.2	52.7	OOM	OOM
4B	4	32.7	33.8	40.4	46.9
4B	8	56.6	59.2	OOM	OOM
12B	2	48.1	53.4	60.7	OOM
12B	4	OOM	OOM	OOM	OOM

Table 30: Peak memory in GiB allocated during prefill on one MI210 at $T{=}1{,}024$, $K{=}2$, and 32 sequences per device, including the weights and excluding the LM head, for the plain forward pass, Jacobi recirculation, and sequential recirculation without compilation. Sequential recirculation keeps its CUDA graphs. All three hold the complete decoding cache at the end, whose size is given in the last column.

Model	Plain	Jacobi	Sequential	Cache
Gemma 3 1B	4.2	4.6	4.6	0.5
Gemma 3 4B	14.6	15.6	15.4	4.3
Gemma 3 12B	38.8	40.4	39.9	12.0
Llama 3.1 8B	23.4	25.0	24.6	4.0
Qwen3 8B	23.8	25.4	25.0	4.5

We time both in every configuration and report the faster, which is up to 28% faster on the A100 and 21% on the H100. Both GPUs are PCIe cards with power limits of 300 and 350 W. The plain forward pass and Jacobi recirculation run at the power limit, and hence at a reduced clock, for most measurements, whereas sequential recirculation stays below the limit at small batches. The speed-ups at small batches are therefore conservative. At $B \geq 32$, sequential recirculation also reaches the power limit in part of each measurement.

Single sequence. Table 31 compares the three GPUs at $T{=}1{,}024$, $B{=}1$, and $K{=}2$. Jacobi recirculation is 45–$98\times$ faster than sequential recirculation on the A100 and 78–$145\times$ on the H100, against 47–$83\times$ on the MI210. The ideal speed-up grows with P/β_{mem}, from about $35\times$ on the MI210 to about $50\times$ on the A100 and 113–$119\times$ on the H100. As on the MI210, the measured speed-up equals the ideal one scaled by $\mathrm{MFU}_{\mathrm{Jac}}/\mathrm{MBU}_{\mathrm{seq}}$. Sequential recirculation uses the bandwidth of the A100 and H100 better than that of the MI210 and reaches 64–74% of peak on the 8B and 12B models. The measured speed-up therefore stays close to the ideal one on the A100 and falls below it on the H100, and the speed-ups on the A100 are close to those on the MI210 except on 1B. On 1B, sequential recirculation attains less than 30% of peak bandwidth on all three GPUs, and the speed-up exceeds the ideal one.

Scaling. Figure 15 shows the speed-ups over the sequence length, the per-device batch, and the number of rounds, and Tables 32 and 33 list their values. On all three GPUs, the speed-up decreases with the per-device batch and with the number of rounds, as the cost model predicts. For the Gemma 3 models, most of whose layers use sliding-window attention, it grows moderately with the sequence length on all three GPUs. Llama 3.1 8B and Qwen3 8B use global attention in every layer, whose cost grows quadratically with the sequence length and is not part of the cost model. Their speed-up at $T{=}16{,}384$ is lower than at $T{=}1{,}024$ on the MI210 and most clearly on the H100, where it falls from $83\times$ and $79\times$ to $63\times$ and $61\times$. The GPUs thus differ mainly in the level of the speed-up and much less in its scaling.

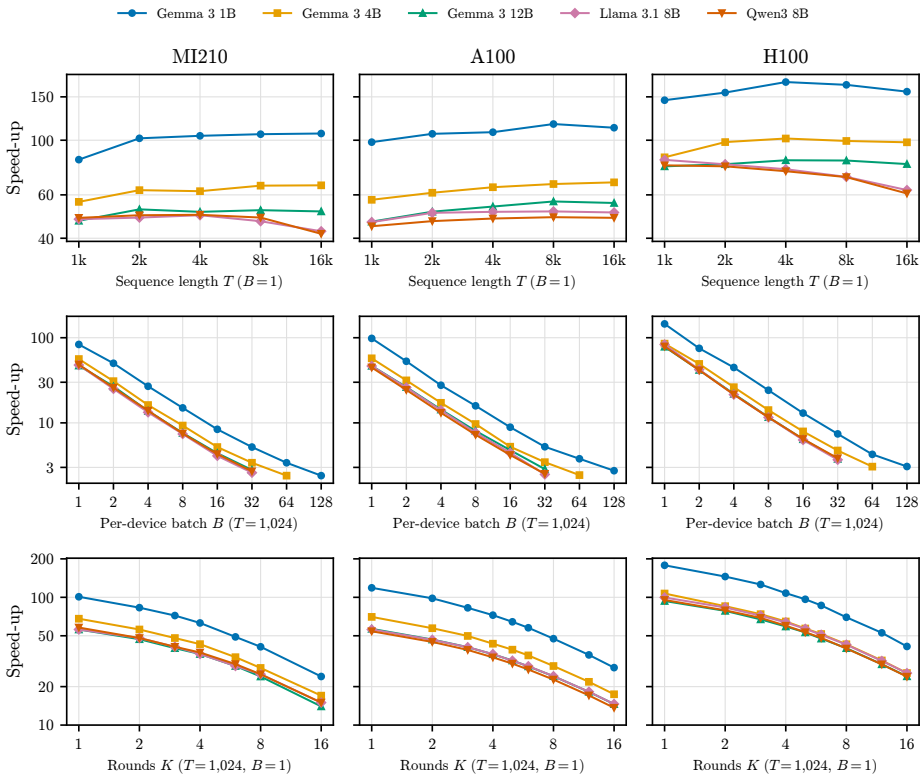


Figure 15: Prefill speed-up of Jacobi recirculation over sequential recirculation on three GPUs in bf16, excluding the LM head, over the sequence length at $B{=}1$ (top), the per-device batch at $T{=}1{,}024$ (middle), and the number of rounds at $T{=}1{,}024$ and $B{=}1$ (bottom). Rows share the vertical axis. $K{=}2$ unless varied. Tables 32 and 33 list the values.

Table 31: Prefill speed-up of Jacobi recirculation over sequential recirculation at $T{=}1{,}024$, $B{=}1$, and $K{=}2$ on three GPUs in bf16, excluding the LM head, with S_{ideal} in parentheses. MFU is the attained fraction of peak compute of Jacobi recirculation and MBU that of peak bandwidth of sequential recirculation, relative to the nominal peaks.

	MI210			A100			H100		
Model	Speed-up	MFU	MBU	Speed-up	MFU	MBU	Speed-up	MFU	MBU
1B	83× (36×)	0.38	0.16	98× (51×)	0.49	0.26	145× (119×)	0.35	0.29
4B	56× (35×)	0.51	0.32	57× (50×)	0.59	0.51	85× (117×)	0.42	0.58
12B	47× (35×)	0.57	0.43	47× (50×)	0.60	0.64	78× (117×)	0.50	0.74
Llama 3.1 8B	48× (34×)	0.60	0.43	47× (49×)	0.61	0.64	83× (114×)	0.53	0.73
Qwen3 8B	48× (34×)	0.58	0.41	45× (48×)	0.61	0.66	79× (113×)	0.50	0.72

Table 32: Prefill speed-up of Jacobi recirculation with $K{=}2$ over sequential recirculation on three GPUs in bf16, excluding the LM head, over the sequence length T at $B{=}1$ and over the per-device batch B at $T{=}1{,}024$, where the column $T{=}$1k is also $B{=}1$. All entries are speed-ups. The MI210 values are those of Table 21. The A100 and H100 use its batch grid, and — marks batches outside it.

		T at $B{=}1$					B at $T{=}1{,}024$						
Model	GPU	1k	2k	4k	8k	16k	2	4	8	16	32	64	128
1B	MI210	83	102	104	106	106	50	27	15	8.4	5.2	3.4	2.4
	A100	98	106	108	116	112	53	28	16	8.9	5.2	3.8	2.7
	H100	145	156	172	168	157	75	45	24	13	7.4	4.3	3.1
4B	MI210	56	63	62	65	66	31	16	9.3	5.2	3.4	2.4	—
	A100	57	61	64	66	67	32	17	9.7	5.2	3.5	2.4	—
	H100	85	98	102	99	98	49	26	14	7.9	4.7	3.1	—
12B	MI210	47	52	51	52	51	27	14	7.6	4.4	2.8	—	—
	A100	47	51	54	56	56	26	14	8.0	4.8	2.9	—	—
	H100	78	80	83	83	80	42	22	12	6.4	3.8	—	—
Llama 3.1 8B	MI210	48	49	50	47	43	25	13	7.4	4.1	2.6	—	—
	A100	47	51	51	51	51	26	14	7.7	4.4	2.5	—	—
	H100	83	80	76	71	63	42	21	12	6.3	3.7	—	—
Qwen3 8B	MI210	48	50	50	49	42	26	14	7.5	4.3	2.7	—	—
	A100	45	47	48	49	48	24	13	7.3	4.2	2.5	—	—
	H100	79	78	75	71	61	42	22	12	6.5	3.8	—	—

Table 33: Prefill speed-up of Jacobi recirculation over sequential recirculation with K rounds at $T{=}1{,}024$ and $B{=}1$ on three GPUs in bf16, excluding the LM head. The time of sequential recirculation does not depend on K. The MI210 values are those of Table 28.

Model	GPU	$K{=}1$	$K{=}2$	$K{=}3$	$K{=}4$	$K{=}6$	$K{=}8$	$K{=}16$
1B	MI210	101×	83×	72×	63×	49×	41×	24×
	A100	119×	98×	83×	73×	58×	47×	28×
	H100	178×	145×	126×	108×	86×	70×	41×
4B	MI210	68×	56×	48×	43×	34×	28×	17×
	A100	70×	57×	50×	43×	35×	29×	17×
	H100	107×	85×	74×	65×	52×	43×	26×
12B	MI210	56×	47×	40×	36×	29×	24×	14×
	A100	57×	47×	41×	36×	29×	24×	15×
	H100	94×	78×	67×	59×	48×	40×	24×
Llama 3.1 8B	MI210	56×	48×	41×	36×	29×	25×	15×
	A100	56×	47×	41×	36×	29×	24×	15×
	H100	100×	83×	72×	64×	51×	43×	26×
Qwen3 8B	MI210	58×	48×	41×	37×	30×	25×	15×
	A100	54×	45×	39×	34×	27×	23×	14×
	H100	95×	79×	69×	60×	48×	40×	24×