LooPiT: Input-Space Specialization for Parallel Loop Transformers

Anonymous authors
Paper under double-blind review

Abstract

Weight-shared recurrence improves language modeling without duplicating the backbone, but sequential loops still execute the full recurrent depth. Cross-loop parallelism (CLP) addresses the decode schedule, but leaves the shared-backbone capacity constraint intact. We study loop-specific capacity under CLP, using Parallel Loop Transformers (PLT) as the substrate, without restoring a full set of unique layers. Alongside the established weight-space route of per-loop adapters, we investigate an input-space route: token-indexed tables private to each layer and loop, with shared projections (S-PLE). On a 135M backbone trained for 100B tokens, S-PLE improves geometric-mean perplexity from 27.50 to 22.91, compared with 26.09 for the tested adapters. A 906M study and two-seed checks support the input-over-weight ordering. Rank and width sweeps test how performance changes with added capacity, while channel ablations examine how each loop uses the learned channels. LooPiT uses approximately 45% fewer backbone parameters than depth-doubled dense models across both scales, with approximately 42% fewer total parameters at the larger scale. On a mobile NPU, the input-only configurations retain a graph decode-latency advantage at both scales. The evidence supports input-space specialization as an effective, compatible alternative to weight adaptation within the tested quality–storage–latency trade-offs; prefill efficiency remains an optimization target.

1 Introduction

Weight sharing offers a simple way to increase a language model's effective depth without storing a separate set of weights for every layer execution. A recurrent model applies one backbone repeatedly, trading additional computation for better quality at a fixed backbone parameter count (Dehghani et al., 2019; Lan et al., 2020; Bae et al., 2025a). Recent layer-sharing and looped language models improve quality without proportionally increasing stored parameters (Liu et al., 2024; Zhu et al., 2025). This makes recurrence attractive for deployment where storing a larger dense model is costly.

Parameter savings, however, do not automatically deliver faster decoding. Two sequential passes through a shared backbone still execute the same number of layers as a depth-doubled dense model. *Cross-loop parallelism* (CLP), introduced in Parallel Loop Transformer (PLT; Wu et al., 2025), addresses this scheduling problem by shifting dependencies so that different loops process different token positions concurrently. This separates the benefits of weight sharing from the cost of a strictly sequential decode schedule.

A second problem remains: *the loops share the same backbone parameters.* Different hidden states permit different behavior, but the loops cannot independently parameterize their transformations. Faster scheduling does not relax this constraint, while assigning a full set of weights to each loop would forfeit the storage benefit of sharing. This motivates the question: *how should loop-specific capacity be added under CLP to improve quality while retaining the benefits of backbone sharing and parallel decoding?*

The established route is through the *weight channel*: give each loop its own low-rank adapter (Bae et al., 2025a). We investigate a different route through the *input channel*: give each (layer, loop) its own learned token-indexed table, read through loop-shared projections. These

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.

shared-projection per-loop embeddings (S-PLE) do not introduce new information—token identity is already available—but provide distinct learned conditioning at each recurrent execution. They can be used alone or alongside weight adapters.

The distinction matters for deployment. Adapter capacity expands matrix operations on hidden states; table capacity expands resident storage while selecting one row per table and token. Stored capacity and active decode work therefore need not grow together. We evaluate this trade-off accounting for table footprints and lookup costs, rather than claiming greater efficiency per stored parameter.

We use *LooPiT* to denote our system with loop-specific input conditioning under CLP, optionally combined with weight adapters. We organize the study around three findings:

1. **Input-space capacity substantially improves quality under CLP.** At 135M/100B, S-PLE32/64 improve gm perplexity from 27.50 to 23.57/22.91, versus 26.09 for rank-8 adapters. The ordering persists across tested rank/width choices and appears at 906M and in unified-recipe two-seed checks (Sections 4 and 4.4).
2. **Input and weight channels can provide complementary specialization.** Combining channels gives small additional likelihood gains in the reported point estimates. Ablations show reliance on both channels; swapping adapters between loops is more disruptive than removing either loop's adapter, indicating dependence on learned loop assignments (Section 5).
3. **Input capacity need not erase the decode advantage.** S-PLE64 and rank-8 adapter graphs have similar measured decode latency at 135M despite different offline quality. At 906M, S-PLE64 retains both total-parameter and graph-latency savings over depth-doubled dense (Section 6).

2 Related Work

Recurrent depth and efficient decoding. The Universal Transformer (Dehghani et al., 2019) and ALBERT (Lan et al., 2020) established weight-shared depth; MobileLLM (Liu et al., 2024) studied block-wise sharing in small language models with tied embeddings (Press and Wolf, 2017). Looped transformers have also been studied as programmable computers (Giannou et al., 2023), algorithm learners (Yang et al., 2024), and billion-parameter latent-reasoning models (Geiping et al., 2025; Zhu et al., 2025). Iso-depth scaling laws (Schwethelm et al., 2026) characterize the quality gap between recurrent and unique-layer depth. PLT (Wu et al., 2025) combines CLP with shared caches and gated local attention to target decode latency. We hold that substrate fixed and study added specialization capacity; learned token-adaptive depth (Bae et al., 2025b) is orthogonal to our fixed-loop-count comparison.

Weight-space loop specialization. Relaxed Recursive Transformers (Bae et al., 2025a) use per-loop LoRA adapters (Hu et al., 2022) when converting pretrained dense models to recurrent form. Mixture-of-LoRAs (Nouriborji et al., 2025) adds token-conditional experts, while RingFormer (Heo et al., 2025) uses input-conditioned low-rank level signals. Our comparison uses static per-loop zFLoRA adapters (Gowda et al., 2025), fused into the backbone projections to avoid separate adapter operations. This is a deployment-oriented representative of the weight channel, not an exhaustive comparison of adapter designs.

Input-space capacity. Per-layer token embeddings are used in Gemma 3n (Gemma Team, 2025) and Gemma 4 (Gemma Team, 2026); indexed memories such as product-key memory (Lample et al., 2019) likewise separate stored capacity from per-token active work. S-PLE applies this principle to a recurrent backbone by making tables private to each *layer and loop*, while sharing their projections across loops. Unlike PLT's reinjection of the shared token embedding, this supplies separately learned conditioning at each execution. Our contribution is this specialization design and its empirical comparison with weight adaptation, not the invention of per-layer embeddings. A dense per-layer-embedding control tests whether the input channel also helps without recurrence.

Supporting training and deployment studies. We use the SmolLM2 architecture/data family (Allal et al., 2025) and WSD scheduling (Hu et al., 2024). Pretrained initialization and distilled cooldowns connect the study to small-model training (Gemma Team, 2024; Busbridge et al., 2025; Muralidharan et al., 2024). Quantization checks address sensitivity highlighted by LoopQ (Fang et al., 2026), but remain separate from our graph-latency measurements (Appendices F, G and K).

3 Loop-Specific Capacity under Cross-Loop Parallelism

Figure 1 separates the shared computation from the two sources of loop-specific capacity. Both channels preserve the shared backbone; they differ in what is made private to each loop and how that capacity is accessed.

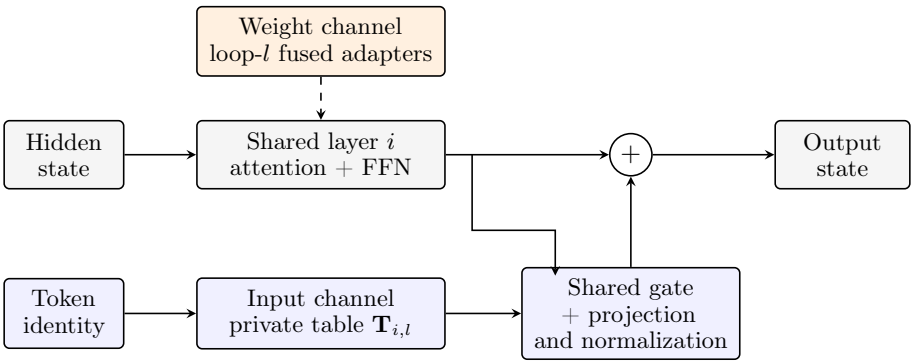


Figure 1: Two compatible specialization channels within one shared layer execution. Weight adapters specialize the shared layer's transformations; S-PLE supplies token-indexed conditioning through private (layer, loop) tables and loop-shared projections. The gate also depends on the shared layer's output. Dashed arrow denotes weights fused into the layer, not a separate inference operation.

3.1 A shared backbone with parallel refinement

With token embeddings $\mathbf{E}$ and backbone f_θ, standard weight-shared recurrence uses $\mathbf{H}^{(0)} = \mathbf{E}$ and $\mathbf{H}^{(l)} = f_\theta(\mathbf{H}^{(l-1)})$, $l = 1, \ldots, L$. Logits are read from the normalized final-loop state. The backbone remains trainable in every condition.

CLP and the PLT substrate. We adopt three mechanisms from Wu et al. (2025) unchanged. Loop 1 takes $\mathbf{E}$ as input and builds a shared post-RoPE KV cache; subsequent loops form fresh queries against that context. Each refinement loop also computes local windowed attention, blended with global attention through a learned per-head sigmoid gate. Finally, PLT refinement adds the original token embeddings to the shifted previous-loop states, using $\mathbf{E} + \text{shift}(\mathbf{H}^{(l-1)})$: position t in loop l reads position $t-1$ in loop $l-1$. This shifted dependency enables CLP. CLP denotes the schedule; PLT denotes the full inherited substrate, including its attention mechanisms. Our plain-loop controls omit the shift, cache sharing, and gated windowed attention, but retain embedding reinjection (Section 4.1). In our comparisons, *PLT, no channels* denotes PLT without S-PLE or loop-specific weight adapters.

3.2 Input channel: shared-projection per-loop embeddings

For vocabulary size $|V|$, each (layer i, loop l) owns a private token table $\mathbf{T}_{i,l} \in \mathbb{R}^{|V| \times d_{\text{ple}}}$. After the shared layer produces $\mathbf{H}_{\text{mid}}$, the channel adds gated token conditioning:

$$\mathbf{p}_{i,l} = \mathbf{T}_{i,l}(\text{input_ids}), \qquad \mathbf{g}_i = \sigma(\mathbf{W}^{(i)}_{\text{gate}} \mathbf{H}_{\text{mid}}), \tag{1}$$

$$\mathbf{H}_{\text{out}} = \mathbf{H}_{\text{mid}} + \text{RMSNorm}\Big(\mathbf{W}^{(i)}_{\text{out}}(\mathbf{g}_i \odot \mathbf{p}_{i,l})\Big). \tag{2}$$

Gate and output projections are shared across loops at each layer; the tables are not. The output projection is zero-initialized, so the channel initially leaves the backbone unchanged. S-PLE augments, rather than replaces, the shared-embedding anchor in PLT refinement.

The design puts loop identity in token-indexed storage while retaining shared dense transformations. For two loops, table storage grows as $n_{\text{layers}}L|V|d_{\text{ple}}$, but each token reads only $n_{\text{layers}}L$ rows, followed by small projections. At 135M, S-PLE64 has 188.7M table and 2.2M projection parameters. Widths 32/64/128 and a private-projection control test choices within this channel (Section 4.4 and appendix C). Tables remain part of the model footprint even when lookup occurs outside the NPU graph.

3.3 Weight channel: loop-specific fused adapters

Each loop j owns a rank-r compander pair (C_j, E_j). Following zFLoRA (Gowda et al., 2025), compressors are concatenated into input-side projections (QKV, gate, up), and expanders into output-side projections (o, down). A per-slot mask selects the active loop's adapter slice. The hardware executes fused base-width+Lr matrix products, without a separate adapter operation. Our primary comparison uses r=8 (2.95M auxiliary parameters at 135M); partial sweeps also test ranks 16 and 32. Fusion details are in Appendix C.

The two channels are structurally compatible: S-PLE changes the token-conditioned residual input to subsequent processing, while the adapters change transformations of hidden states. We train each alone and together on the same PLT backbone. S-PLE32/64 use 32–65× the auxiliary parameters of rank-8 adapters; the comparison therefore tests practical channel configurations, not equal-capacity mechanisms.

4 Does Input-Space Capacity Improve Quality?

4.1 Comparison design

The primary backbone is a SmolLM2-family decoder with d=576, 30 layers, 9 query / 3 KV heads, tied embeddings, and $|V|$=49,152. A two-loop model executes this backbone twice, matching the 60 layer-executions of the depth-doubled dense reference while storing 134.5M rather than 241M backbone parameters. Equal executed depth does not imply equal FLOPs.

The primary study trains all scratch conditions on the same 100B-token SmolLM2 corpus, with sequence length 2048, an effective batch of ≈1.05M tokens, and AdamW with WSD scheduling. The backbone, data order, and optimization recipe are fixed across channel arms. A second study uses a 906M backbone and 50B tokens. Full training settings and run provenance are in Appendix A.

We compare same-backbone dense, plain-loop controls, channel-free PLT, each channel separately, and their combination; depth-doubled dense provides a unique-layer reference. Our plain-loop controls use unshifted recurrence with additive embedding reinjection; this is a control choice, not a requirement of standard recurrence. Exact initialization is documented in Appendix A.

The primary likelihood metric is the geometric mean of LAMBADA and WikiText harness perplexity (*gm ppl*). Training curves and the second-scale table use WikiText-2 token-level validation perplexity instead; these metrics are not interchangeable. Downstream evaluation reports nine 0-shot accuracy tasks and CoQA F1. We use *avg-7*, excluding near-chance ARC-c and Winogrande, as the compact accuracy summary; full task scores are in Table 5. Generative evaluation uses the train-consistent refinement path (Appendix I).

4.2 Input capacity delivers the larger single-channel gain

Table 1 compares the 135M models after 100B training tokens.

Scheduling alone does not close the quality gap. The plain-loop control improves gm ppl from 29.55 to 27.02 without duplicating backbone weights. PLT without channels reaches

Table 1: 135M-backbone study at 100B tokens, all trained from scratch. Aux includes tables and projections; gm ppl is the geometric mean of LAMBADA and WikiText perplexity. avg-7 excludes two near-chance tasks. These are single-seed endpoints; seed checks are discussed below and full task scores are in Table 5.

Model	Aux (M)	gm ppl↓	avg-7↑
Dense, same backbone	—	29.55	52.46
Plain-loop control, two loops	—	27.02	53.70
PLT, no channels	—	27.50	52.10
PLT + weight ($r = 8$)	2.95	26.09	52.99
PLT + input ($d_{\text{ple}} = 32$)	95.5	23.57	54.10
PLT + input ($d_{\text{ple}} = 64$)	191.0	22.91	54.86
PLT + input ($d_{\text{ple}} = 128$)	382.0	24.13	54.34
PLT + both ($r = 8$, $d_{\text{ple}} = 64$)	194.0	22.80	54.63
Dense, depth-doubled	—	22.04	56.07

27.50: its scheduling machinery does not itself recover the quality of depth-doubled dense (22.04). This separates the motivation for CLP from the capacity problem addressed here.

Input conditioning provides the larger improvement. Relative to channel-free PLT, rank-8 adapters improve gm ppl by 1.41, while S-PLE32 and S-PLE64 improve it by 3.93 and 4.59. S-PLE64 also improves avg-7 from 52.10 to 54.86, compared with 52.99 for the weight-only arm. The result is not a uniform task-wise win, and dense depth-doubling remains better on both summaries. Nevertheless, the input channel moves PLT substantially closer to the unique-layer reference without duplicating the backbone.

The channels can help together, but the final increment is small. Adding rank-8 adapters to S-PLE64 gives 22.80 gm ppl, compared with 22.91 for S-PLE64 alone. This is a small unreplicated increment, not a demonstrated statistical interaction or a consistent accuracy gain: avg-7 is slightly lower for the combined arm.

Input tables are useful beyond recurrence. A separate dense + S-PLE64 control improves gm ppl from 29.55 to 26.67 (Appendix B.1, Table 5). Its approximately 97M auxiliary parameters differ from the recurrent arm's 191M because it has one table per layer rather than per (layer, loop). This supports the input channel's broader utility; subtracting the dense and recurrent gains would not isolate a loop-specific causal effect.

4.3 Does the ordering persist across scale and seeds?

Table 2 summarizes the second study, which uses a 12-layer, 906M backbone (d=2048, 32 query and KV heads, FFN width 8192, tied embeddings) trained for 50B tokens. Two loops match the 24 layer-executions of the 1.71B dense reference. S-PLE64 adds 78.7M parameters and rank-8 adapters add 6.3M.

The input-only arm again has better likelihood than weight-only (13.86 vs. 14.17); the combination reaches 13.80 at width 64. Bare recurrence improves on same-parameter dense, and the added channels further improve quality while remaining substantially smaller than depth-doubled dense. Unique layers still give the best validation perplexity. Extended web/code and generative evaluations support a broader comparison but not a universal task ordering (Appendix H). Since architecture, token budget, and perplexity metric differ between studies, we do not infer a scaling law.

We also retrained S-PLE64 and rank-8 adapters at 135M for 100B tokens with two seeds under a modified recipe without dropout. S-PLE64 achieved lower gm perplexity in both runs, by 1.82 and 2.47 points (Appendix D).

Table 2: Second-scale study: 906M backbone, 50B tokens. All recurrent rows execute 24 layers; same-backbone dense executes 12. Quality is token-level validation perplexity, not the gm ppl of Table 1. Full endpoints and intermediate checkpoints are in Table 6.

Model	Total params (M)	val ppl↓
Dense, same backbone	906	14.66
Plain-loop control, two loops	906	14.20
PLT + weight ($r = 8$)	912	14.17
PLT + input ($d_{\text{ple}} = 64$)	985	13.86
PLT + both ($r = 8$, $d_{\text{ple}} = 64$)	991	13.80
PLT + both ($r = 8$, $d_{\text{ple}} = 128$)	1,070	13.71
Dense, depth-doubled	1,711	12.73

4.4 How sensitive is the comparison to channel size?

The larger input-channel gain accompanies a much larger table budget. To test whether the practical ordering depends on a single rank/width choice, we vary capacity *within* each channel. At 135M, S-PLE widths 32/64/128 reach 23.57/22.91/24.13 gm ppl: doubling width beyond 64 does not help in this recipe. Sharing projections across loops also performs comparably to private projections at width 64 (a 0.12 validation-ppl point-estimate advantage; Appendix C), supporting the shared-projection design.

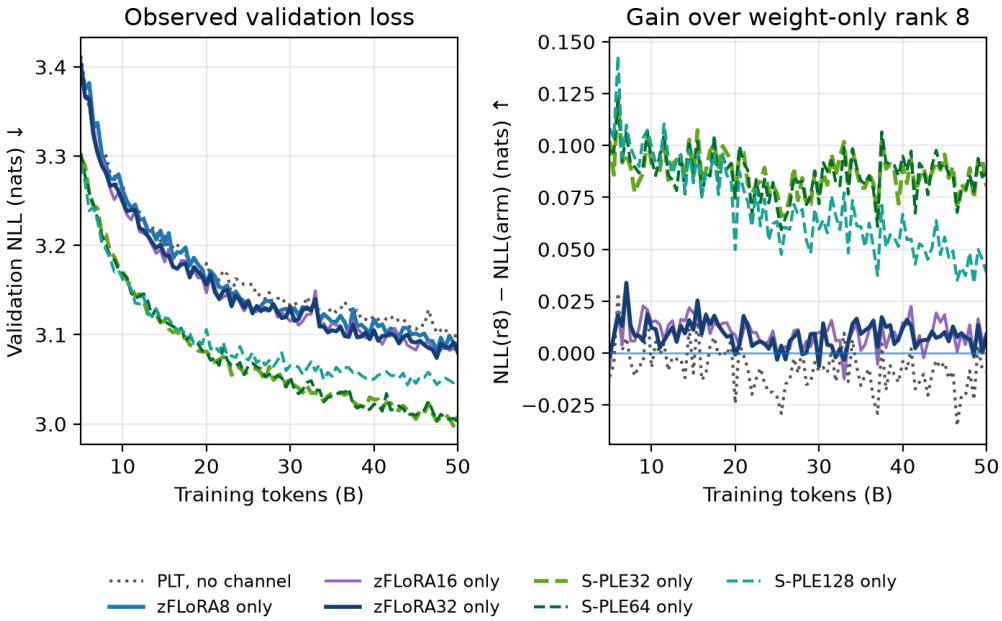


Figure 2: Weight-only rank sensitivity with input-channel references. Validation NLL from 5B to 50B training tokens for weight-only ranks 8/16/32, channel-free PLT, and S-PLE32/64/128 on the 135M backbone. Right: NLL reduction relative to rank 8 in the same sweep. Higher adapter rank helps modestly over this interval without closing the observed input-channel gap. These are partial trajectories, not extrapolated endpoints or capacity-matched comparisons (Appendix J.1).

In Figure 2, weight-only ranks 8, 16, and 32 reach validation NLL 3.0893, 3.0844, and 3.0804, respectively, after 50B training tokens. Increasing rank from 8 to 32 reduces NLL by 0.0089, while S-PLE32 remains 0.0715 lower than rank 32. All three S-PLE widths have lower NLL than all three tested weight-only ranks at this checkpoint. Rank 32 has approximately 11.8M

auxiliary parameters versus S-PLE32's 95.5M; these partial trajectories do not establish equal-capacity effects or final-100B ordering.

Conditional on S-PLE64, a separate rank sweep gives rank 32 a 0.0058 NLL advantage over rank 8 at 25B. At 906M, doubling S-PLE width from 64 to 128 with rank 8 fixed improves validation ppl from 13.80 to 13.71 (Figure 6 and appendix E).

What the sweeps establish. The sweeps extend the practical comparison beyond one rank/width choice: larger adapters do not close the observed gap, while larger input tables are not uniformly better. They do not isolate placement from size.

5 What Do the Specialization Channels Learn?

The channel comparison establishes a quality ordering at the tested budgets. We next ask whether the learned channels are actually used and whether their assignments to loops matter. Zeroing and shuffling interventions on saved checkpoints provide more direct evidence of dependence than representation similarity alone (Appendix E and Table 3).

Table 3: Dependence on trained channel assignments at 135M/100B. Entries are increases in NLL after intervention on 8 validation sequences (2048 tokens each); larger is more disruptive, not a larger training-time benefit. Zero removes one loop's channel; swap exchanges adapters between loops. Subset probes are distinct from trainer-parity validation scores. Full protocols and full-validation S-PLE checks are in Appendix E.

	Zero S-PLE		Zero adapters		Swap
Trained channels	loop 1	loop 2	loop 1	loop 2	adapters
Input only	7.48	1.90	—	—	—
Weight only	—	—	6.19	3.53	11.26
Both	7.77	1.86	2.00	0.24	7.87

The model uses token-conditioned capacity in both loops. Zeroing S-PLE in either loop increases NLL for both the input-only and combined 135M models (Table 3). Full-validation interventions on the 906M combined model likewise give positive costs in both loops: 0.768 ± 0.013 and 0.265 ± 0.011 (bootstrap 95% CI half-widths; Tables 11 and 12). Shuffling table rows also hurts. These results support learned token-dependent conditioning, with greater ablation sensitivity in the first loop.

Weight specialization remains useful alongside input conditioning. In the 135M combined model, zeroing either loop's adapters increases NLL, and exchanging their assignments is more disruptive (Table 3). Thus input conditioning does not make the trained adapters interchangeable or unused. The channels can participate in a co-adapted solution even when their combined endpoint gain over S-PLE alone is small. Switching off either channel makes the trained model perform worse, showing that it has learned to use both. But this does not explain why S-PLE gives better results than adapters when each is trained separately.

6 Which Resource Advantages Are Retained?

The deployment objective is better quality under CLP without paying for another full set of dense layers. We distinguish three resources: shared-backbone parameters, total resident parameters including tables, and measured graph decode latency. Weight traffic motivates this distinction on constrained devices (Alizadeh et al., 2024), but timing alone does not establish on-chip residency or DRAM traffic. Table 4 compares these resources alongside offline quality at both scales.

Table 4: Quality–storage–latency operating points. Quality is measured offline in floating point (135M: gm ppl; 906M: token validation ppl). Latency is measured separately on a Galaxy S26 Ultra NPU with w4a16-kv8 graphs, excluding host-side lookup and application work. Total parameters include token tables. Quantized on-device quality is unmeasured.

Backbone	Model	Params (M)	ppl↓	TPOT (ms)↓
135M	Dense, same backbone	134.5	29.55	2.89
	PLT, no channels	134.5	27.50	3.72
	PLT + weight ($r = 8$)	137.5	26.09	4.52
	PLT + input (width 64)	325.5	22.91	4.55
	PLT + both	328.5	22.80	5.38
	Dense, depth-doubled	241.0	22.04	5.05
906M	Dense, same backbone	906	14.66	10.08
	PLT, no channels	906	—	10.89
	PLT + weight ($r = 8$)	912	14.17	11.87
	PLT + input (width 64)	985	13.86	11.37
	PLT + both (width 64)	991	13.80	11.86
	Dense, depth-doubled	1,711	12.73	19.12

The input channel retains a measured decode advantage. At 135M, channel-free PLT decodes at 3.72 ms/token versus 5.05 for depth-doubled dense. S-PLE64 raises this to 4.55, retaining a smaller graph-latency advantage while improving offline likelihood. The weight-only graph takes 4.52 ms/token: its 0.03 ms difference from S-PLE64 is below the observed 0.17 ms rebuild variation. At 906M, S-PLE64 takes 11.37 ms/token versus 19.12 for depth-doubled dense, with 985M rather than 1,711M total parameters. Relative to channel-free PLT, S-PLE64 adds about 22% decode time at 135M and 4% at 906M. These are different-quality operating points, not quality-matched speedups.

Storage savings depend on the table budget. The shared backbone is retained at both scales, but total parameter savings are not automatic. At 135M, S-PLE64's 325.5M parameters exceed depth-doubled dense's 241M; its advantage is the measured decode/quality trade-off, not a smaller total model. At 906M, the same width preserves total-parameter savings as well. A detailed memory budget is provided in Appendix K, Table 23.

Parameter matching and inference cost. Equal stored capacity need not imply equal inference costs. Matching S-PLE32's 95.5M auxiliary parameters with adapters would bring the 135M-backbone model to approximately 230M total, near the 241M depth-doubled dense reference. Unlike token-indexed tables, larger adapters expand matrix operations executed at every token. This could erode the efficiency motivating backbone sharing; these high-rank configurations remain unmeasured.

Combining channels is not always the best operating point. At 135M, the combined graph takes 5.38 ms/token, exceeding depth-doubled dense for a small likelihood improvement over S-PLE alone. At 906M it takes 11.86 ms/token, retaining the dense-reference latency advantage. Thus compatibility of the channels does not imply that both should always be deployed; their value depends on the quality, storage, and latency budget.

Prefill remains an optimization target. The current NPU export has slower prefill than dense (Table 22). At 906M, S-PLE64 processes the prompt at 4.5k tok/s, versus 14.9k for same-backbone dense and 7.0k for depth-doubled dense. For the measured 1536-token prompt, this corresponds to approximately 0.34 s of prefill; the estimated full graph workload with 512 output tokens is 6.16 s, versus 10.01 s for depth-doubled dense. Decode savings therefore outweigh the prefill penalty for *this workload*, but do not remove its first-token-latency cost. Longer prompts or shorter outputs can change that balance. Careful profiling and optimization of the prefill schedule and kernels are necessary; the current measurements do not establish

how much overhead is removable. Nor do they establish a cloud-serving penalty: GPU kernels, batching, and workload distributions require a separate, matched prefill study.

Measurement boundary. The full 2×2 channel ablation uses one handset and export recipe (Appendix K). Timings cover graph calls, not host embedding/RoPE preparation, table lookup, sampling, or complete application latency. Calibration uses dummy embedding inputs, so the graphs do not establish quantized model accuracy. Separate RTN/GPTQ simulations show quantizer-dependent outcomes, not validation of this handset path (Appendix G). The quality–latency visualization likewise pairs separately measured quantities (Figure 7).

7 Limitations

Our study evaluates input and weight specialization on the PLT substrate at two backbone scales and fixed training budgets. The primary comparisons use one seed; additional two-seed experiments use a modified recipe without dropout. The channels differ in auxiliary parameter count, including in the dense input control, so the comparisons characterize their observed quality and resource trade-offs rather than effects at equal capacity. The weight-only adapter-rank sweep covers training through 50B tokens, and the small combined-channel gains warrant further replication. Downstream improvements vary by task.

Deployment measurements cover graph execution on one mobile NPU. End-to-end latency, host-side lookup costs, and quantized on-device quality require further evaluation. Pre-trained initialization and distillation involve additional teacher computation and are reported separately in Appendices B.3 and F, respectively.

8 Conclusion

Weight sharing, parallel decoding, and loop-specific capacity address different constraints. Recurrence increases effective depth while reusing backbone parameters; CLP allows different loops to execute concurrently during decoding. LooPiT adds learned input conditioning to this shared computation. Its S-PLE channel gives each layer and loop a private token-indexed table, read through loop-shared projections, so that successive executions can specialize without duplicating the backbone. This channel can be used alone or combined with weight adapters.

In the 135M study, S-PLE64 improves gm perplexity from 27.50 to 22.91, compared with 26.09 for rank-8 adapters. The input-over-weight ordering also appears at 906M and in the two-seed study. Increasing adapter rank helps over the observed training interval, but does not close the gap to the input-channel references. Combining the channels gives a small additional quality improvement, and ablations show that the trained model uses both. Exchanging adapters between loops further shows that their learned assignments matter, even when the combined model's advantage over S-PLE alone is small.

The deployment results explain why stored capacity and active computation should be considered together. S-PLE expands token-indexed storage, while higher adapter ranks expand matrix operations at every token. At 135M, S-PLE64 and rank-8 adapters have similar measured graph decode latency despite their different offline quality and parameter counts. Input-only S-PLE retains a graph decode advantage over depth-doubled dense at both scales; at 906M, it also uses approximately 42% fewer total parameters. The useful operating point therefore depends on the available storage as well as the quality and latency targets.

Prefill efficiency is an immediate optimization target. Further work should profile the prefill schedule and kernels, account for host-side table lookup, and evaluate quantized quality in complete application workloads. These measurements can guide the choice of table width, adapter rank, and loop count for a given device. LooPiT establishes loop-specific input conditioning as an effective way to improve the tested recurrent models while retaining backbone sharing and the decode benefits of CLP.

Reproducibility statement

The model and training setup, evaluation protocol, and capacity census are documented in Sections 3 and 4 and appendix C. Recorded seeds, checkpoint-resume behavior, and task-metric mappings are detailed in Appendix A. An anonymized repository for review will contain code, runner scripts, raw evaluation outputs, and the script that regenerates the results table.

AI use statement

AI assistants were used for code refactoring, evaluation-infrastructure debugging, manuscript editing, literature and consistency checks, figure preparation, feedback on experimental comparisons and result interpretation, and internal adversarial review during drafting. Experimental measurements come from the reported training and evaluation runs, not generated AI outputs. The authors are responsible for validating all code, analyses, citations, and scientific claims.

References

K. Alizadeh et al. LLM in a flash: Efficient large language model inference with limited memory. In *ACL*, 2024. https://aclanthology.org/2024.acl-long.678/.

L. B. Allal et al. SmolLM2: When smol goes big — data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025. https://arxiv.org/abs/2502.02737.

S. Bae, A. Fisch, H. Harutyunyan, Z. Ji, S. Kim, and T. Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise LoRA. In *ICLR*, 2025. https://openreview.net/forum?id=WwpYSOkkCt.

S. Bae et al. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. *arXiv preprint arXiv:2507.10524*, 2025. https://arxiv.org/abs/2507.10524.

Y. Bisk, R. Zellers, R. Le Bras, J. Gao, and Y. Choi. PIQA: Reasoning about physical commonsense in natural language. In *AAAI*, 2020. https://arxiv.org/abs/1911.11641.

D. Busbridge, A. Shidani, F. Weers, J. Ramapuram, E. Littwin, and R. Webb. Distillation scaling laws. In *ICML*, 2025. https://proceedings.mlr.press/v267/busbridge25a.html.

C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*, 2019. https://aclanthology.org/N19-1300/.

P. Clark et al. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. https://arxiv.org/abs/1803.05457.

M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser. Universal transformers. In *ICLR*, 2019. https://arxiv.org/abs/1807.03819.

R. Fang, H.-W. Chen, and M.-S. Chen. LoopQ: Quantization for recursive transformers. *arXiv preprint arXiv:2605.16343*, 2026. https://arxiv.org/abs/2605.16343.

E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *ICLR*, 2023. https://arxiv.org/abs/2210.17323.

L. Gao et al. The language model evaluation harness. *Zenodo*, 2024. https://doi.org/10.5281/zenodo.12608602.

J. Geiping et al. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *NeurIPS*, 2025. https://arxiv.org/abs/2502.05171.

Gemma Team. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024. https://arxiv.org/abs/2408.00118.

Gemma Team. Gemma 3n model overview. Google DeepMind developer documentation, 2025. https://ai.google.dev/gemma/docs/gemma-3n.

Gemma Team. Gemma 4 model card. Google DeepMind developer documentation, 2026. https://ai.google.dev/gemma/docs/core/model_card_4.

A. Giannou, S. Rajput, J.-Y. Sohn, K. Lee, J. D. Lee, and D. Papailiopoulos. Looped transformers as programmable computers. In *ICML*, 2023. https://proceedings.mlr.press/v202/giannou23a.html.

D. Gowda, S. Song, H. Goka, and J. Lee. zFLoRA: Zero-latency fused low-rank adapters. In *EMNLP*, 2025. https://aclanthology.org/2025.emnlp-main.1086/.

J. Heo, E. Fozilov, H. Song, and T. Kim. RingFormer: Rethinking recurrent transformer with adaptive level signals. In *Findings of EMNLP*, 2025. https://aclanthology.org/2025.findings-emnlp.1182/.

K. M. Hermann et al. Teaching machines to read and comprehend. In *NeurIPS*, 2015. https://arxiv.org/abs/1506.03340.

E. J. Hu et al. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022. https://arxiv.org/abs/2106.09685.

S. Hu et al. MiniCPM: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024. https://arxiv.org/abs/2404.06395.

S. Kornblith, M. Norouzi, H. Lee, and G. Hinton. Similarity of neural network representations revisited. In *ICML*, 2019. https://proceedings.mlr.press/v97/kornblith19a.html.

G. Lample, A. Sablayrolles, M. Ranzato, L. Denoyer, and H. Jégou. Large memory layers with product keys. In *NeurIPS*, 2019. https://arxiv.org/abs/1907.05242.

Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *ICLR*, 2020. https://arxiv.org/abs/1909.11942.

C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, 2004. https://aclanthology.org/W04-1013/.

J. Lin et al. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *MLSys*, 2024. https://arxiv.org/abs/2306.00978.

Z. Liu et al. MobileLLM: Optimizing sub-billion parameter language models for on-device use cases. In *ICML*, 2024. https://proceedings.mlr.press/v235/liu24ce.html.

S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models. In *ICLR*, 2017. https://arxiv.org/abs/1609.07843.

T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *EMNLP*, 2018. https://aclanthology.org/D18-1260/.

S. Muralidharan et al. Compact language models via pruning and knowledge distillation. *arXiv preprint arXiv:2407.14679*, 2024. https://arxiv.org/abs/2407.14679.

M. Nouriborji, M. Rohanian, and O. Rohanian. Improving recursive transformers with mixture of LoRAs. *arXiv preprint arXiv:2512.12880*, 2025. https://arxiv.org/abs/2512.12880.

D. Paperno et al. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *ACL*, 2016. https://aclanthology.org/P16-1144/.

O. Press and L. Wolf. Using the output embedding to improve language models. In *EACL*, 2017. https://aclanthology.org/E17-2025/.

S. Reddy, D. Chen, and C. D. Manning. CoQA: A conversational question answering challenge. *TACL*, 7:249–266, 2019. https://aclanthology.org/Q19-1016/.

K. Sakaguchi, R. Le Bras, C. Bhagavatula, and Y. Choi. WinoGrande: An adversarial Winograd schema challenge at scale. In *AAAI*, 2020. https://arxiv.org/abs/1907.10641.

K. Schwethelm, D. Rückert, and G. Kaissis. How much is one recurrence worth? Iso-depth scaling laws for looped language models. *arXiv preprint arXiv:2604.21106*, 2026. https://arxiv.org/abs/2604.21106.

J. Welbl, N. F. Liu, and M. Gardner. Crowdsourcing multiple choice science questions. In *WNUT*, 2017. https://aclanthology.org/W17-4413/.

B. Wu et al. Parallel loop transformer for efficient test-time computation scaling. *arXiv preprint arXiv:2510.24824*, 2025. https://arxiv.org/abs/2510.24824.

L. Yang, K. Lee, R. Nowak, and D. Papailiopoulos. Looped transformers are better at learning learning algorithms. In *ICLR*, 2024. https://arxiv.org/abs/2311.12424.

R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. HellaSwag: Can a machine really finish your sentence? In *ACL*, 2019. https://aclanthology.org/P19-1472/.

R.-J. Zhu et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025. https://arxiv.org/abs/2510.25741.

A Run and evaluation reproducibility

Plain-loop control convention. The reported plain-loop controls enable additive input mixing: $\mathbf{H}^{(0)} = \mathbf{E}$ and $\mathbf{H}^{(l)} = f_\theta(\mathbf{E} + \mathbf{H}^{(l-1)})$ for $l = 1, \ldots, L$, with final normalization before the output head. Thus the first backbone pass receives $2\mathbf{E}$; PLT instead starts from $\mathbf{E}$ and adds embeddings to shifted states only on later loops. The controls use ordinary causal attention without PLT's position shift, shared KV cache, or gated windowed attention. They are unshifted, embedding-reinjected controls, distinct from the standard recurrence defined in Section 3.1.

Primary training and evaluation recipe. The scratch study uses the identical SmolLM2 100B-token raw corpus (EOS-packed, on-the-fly tokenization), sequence length 2048, effective batch ≈1.05M tokens per step, AdamW (β_1=0.9, β_2=0.95), peak LR 3×10^{-3} with WSD (2,000 warmup steps, 20% linear decay tail), weight decay 0.01, clip 1.0, label smoothing 0.1, dropout 0.1, bf16, and DeepSpeed ZeRO-0 on 16×H100. The local attention window is 128, with the sigmoid mixing gate initialized to approximately 0.12. Final checkpoints are evaluated at max length 2048, 0-shot unless specified: HellaSwag (Zellers et al., 2019), ARC-e/c (Clark et al., 2018), PIQA (Bisk et al., 2020), Winogrande (Sakaguchi et al., 2020), OpenBookQA (Mihaylov et al., 2018), BoolQ (Clark et al., 2019), SciQ (Welbl et al., 2017), LAMBADA accuracy (Paperno et al., 2016), and CoQA F1 (Reddy et al., 2019). The nine-task accuracy average excludes CoQA; avg-7 additionally excludes ARC-c and Winogrande. Validation perplexity is exponential token-level cross-entropy on WikiText-2 (Merity et al., 2017), distinct from the harness word-perplexity measure.

Second-scale recipe and run history. The 906M study uses the same corpus, WSD with peak LR 3×10^{-4}, and a 50B-token budget. All arms share per-layer shapes (d=2048, 32 query / 32 KV heads, FFN 8192, $|V|$=49,152, tied embeddings). Seven of eight planned arms completed. The weight-only run (A2) resumed from its 17.5B rolling checkpoint after an infrastructure failure, retaining the original schedule; a pristine rerun is unavailable. S-PLE256 (A4) aborted at launch twice and supplies no trained endpoint. The run-series alias 840m refers to the actual 906M backbone, not a separate model scale.

Checkpoints and reproducibility. All experiments run from a single repository with per-experiment runner scripts and fixed data order (shard-level shuffling with recorded seeds). The checkpoint policy hard-saves at 25B tokens and at the pre-decay boundary (80% of the target token budget), in addition to 5B-token rolling snapshots. A run can be extended by resuming from its pre-decay checkpoint with a re-planned WSD horizon: resume restores the optimizer and data cursor and fast-forwards the learning-rate schedule. This path was verified by an automated two-phase smoke test. Evaluation uses `lm-eval` task definitions (Gao et al., 2024) with recorded task-metric mappings; the results table is regenerated by one script from raw per-task JSON. Code, runner scripts, raw evaluation outputs, and the table generator will be released in an anonymized repository for review.

B Full quality results and initialization condition

B.1 Primary benchmark results

The complete 135M-family evaluation includes 12 scratch conditions and one pretrained-initialization condition at 100B tokens, plus the official SmolLM2-135M checkpoint as an external reference. The compact scratch-channel comparison in Table 1 selects the rows needed for the main research question; Table 5 retains all tasks and the three-loop controls. The official model's gm ppl of 21.00 versus our scratch dense baseline's 29.55 reflects different training histories and is not a matched-budget comparison.

Table 5: 100B-token final checkpoints, 0-shot. **avg** = mean of the nine accuracy tasks; **avg-7** drops ARC-c and Winogrande; **gm ppl** = geometric mean of LAMBADA and WikiText perplexity. Italics = dense 240M reference. No per-column best is marked: columns are single-seed and uncorrected for multiplicity (seed spans in Appendix D).

Model	hs	arc_e	arc_c	piqa	wino	obqa	boolq	sciq	lamb	avg	gm ppl↓	avg-7	coqa
chance floor	*25*	*25*	*25*	*50*	*50*	*25*	*50*	*25*	*≈0*	—	—	—	—
hf-135m (official)	43.0	58.8	29.6	68.3	52.8	32.8	60.5	84.1	43.0	52.5	21.00	55.79	38.2
base-135m (ours)	39.2	47.1	25.7	67.7	52.0	32.2	62.0	80.0	39.0	49.4	29.55	52.46	34.0
base-135m-ple64	41.1	48.8	25.8	67.4	52.7	30.8	61.6	81.7	41.0	50.1	26.67	53.20	36.7
base-240m (dense 2× depth)	*45.8*	*53.9*	*28.6*	*69.2*	*53.0*	*34.4*	*59.7*	*84.1*	*45.4*	*52.7*	*22.04*	*56.07*	*42.4*
loop-l2	41.1	48.9	25.7	67.8	52.7	32.8	61.7	82.9	40.7	50.5	27.02	53.70	37.9
loop-l3	41.8	48.5	25.9	67.7	52.4	31.4	58.8	80.2	40.2	49.6	26.86	52.66	39.5
loopit-l2	39.7	47.3	26.3	66.1	51.5	32.6	57.8	81.0	40.2	49.2	27.50	52.10	36.1
loopit-l2-sp32	42.3	49.8	27.0	68.7	49.9	32.6	58.2	82.8	44.3	50.6	23.57	54.10	39.2
loopit-l2-sp64	43.0	51.3	27.2	68.6	53.1	32.2	61.5	83.7	43.7	51.6	22.91	54.86	40.3
loopit-l2-sp128	42.0	50.0	27.4	68.7	51.7	32.0	60.5	83.8	43.4	51.1	24.13	54.34	37.6
loopit-l2-zf8	40.5	47.7	26.7	67.6	51.9	32.0	58.5	83.0	41.6	49.9	26.09	52.99	35.2
loopit-l2-zf8-sp64	43.7	50.1	28.2	69.6	52.2	34.0	57.4	83.8	43.8	51.4	22.80	54.63	39.4
loopit-l2-zf8-sp64-pi	45.2	51.3	28.6	68.4	53.6	32.4	52.9	85.2	45.9	51.5	21.51	54.47	44.2
loopit-l3	41.4	46.6	26.9	67.9	54.0	31.4	59.8	81.8	42.1	50.2	25.73	53.00	34.4

B.2 Second-scale endpoints and intermediate checkpoints

Table 6 preserves the run aliases and intermediate measurements behind Table 2. At 17.5B, the S-PLE arms are 0.45–0.54 validation-ppl ahead of the weight-only arm, which is 0.28 ahead of the plain-loop control. At 50B, the weight-only–plain-loop gap is 0.03 ppl, a small unreplicated difference rather than evidence of equivalence. The best looped point remains 0.98 ppl behind depth-doubled dense. Extended task results appear in Appendix H.

B.3 Pretrained initialization as a supporting condition

The preinitialized variant changes nothing at inference: the trunk is initialized from the official pretrained SmolLM2-135M (∼2T tokens; the shared trunk serves every loop by construction) while the loop-specific channels — companders, S-PLE tables, gates — start fresh and the 100B recipe runs unchanged, following the dense→recurrent conversion of Bae et al. (2025a). It inherits the teacher's data exposure and optimization, so it is not a causal test of recurrence stabilization; we report it as an initialization condition (Appendix B.3).

Table 6: 906M-backbone study at 50B tokens: final validation perplexity (wikitext, shift-by-1 CE). Layer-exec = individual layer executions per token; Aux = auxiliary channel parameters. The @17.5B column is the matched-token comparison at equal training progress; A2's @17.5B entry is its pre-loss checkpoint (resumed to target thereafter).

Arm	Params	Aux	Layer-exec	val ppl @50B	@17.5B
T2 base-1p7b (dense 24L)	1,711M	—	24	**12.73**	16.40
A3 `zf8+sp128`	1,070M	163.7M	24	**13.71**	17.21
T3 `zf8+sp64`	991M	85.0M	24	13.80	17.24
A1 `sp64`	985M	78.7M	24	13.86	17.30
A2 `zf8` (weight-only)	912M	6.3M	24	14.17	17.75
L6 looped-L2 plain	906M	—	24	14.20	18.03
T1 base-840m (dense 12L)	906M	—	12	14.66	18.70

Quality and loop alignment. Pre-initializing the trunk from the 2T-token SmolLM2-135M teacher (identical `zf8`+S-PLE recipe and 100B run) is worth 0.37 ppl over its scratch sibling (wikitext val 17.11 vs. 17.48) and is the only 100B-census run to pass the dense-240M baseline (17.35; the 45B cooldown twins of Appendix F.1 also pass it). Under preinit, ablating S-PLE tables is less costly (+0.35–1.0 nll vs. +1.9–7.8 for scratch) and the loops' representations are more aligned (mean uncentered linear kernel alignment 0.69 vs. 0.48–0.51; metric defined in Appendix E). These observations are consistent with greater reliance on the pretrained trunk, but do not isolate optimization dynamics from the starting checkpoint or identify which component carries token-conditioned refinement.

The preinitialized model's 21.51 gm ppl is lower than the depth-doubled reference's 22.04, while CoQA F1 is 44.2 versus 42.4. HellaSwag is 0.6 points lower, ARC-c is equal, and avg-7 remains lower (54.47 versus 56.07); its likelihood advantage is not a general accuracy advantage. Small per-task leads are single-seed and uncorrected for multiplicity.

C Channel cost census

Fused weight-channel construction. The loop-specific compressors and expanders augment the shared matrices as

$$\mathbf{W}_C = [\,\mathbf{W} \mid C_0 \mid \cdots \mid C_{L-1}\,], \qquad \mathbf{W}_E = \begin{bmatrix} \mathbf{W}' \\ E_0 \\ \vdots \\ E_{L-1} \end{bmatrix}. \tag{3}$$

A per-slot binary mask selects the active loop's slice of the augmented residual stream. This keeps adaptation inside the matrix products rather than adding a separate adapter operation.

Table 7 lists representative auxiliary capacity channels of the 135M fleet with its parameter count, streamed-vs-gathered accounting, and inference primitive; Table 8 is the arm ledger behind Table 5. Historical run aliases retain the `loopit` prefix even for PLT controls without S-PLE; these aliases identify experiments, not the scope of the LooPiT system name.

Projection-sharing control. At S-PLE64, private per-(layer, loop) projections use 193.2M auxiliary parameters, versus 191.0M with loop-shared projections, and give a 0.12 higher validation-perplexity point estimate. The tables remain private in both conditions; this comparison does not test sharing the token tables. The width sweep gives validation ppl 17.94/17.77/18.23 at widths 32/64/128. The main channel comparisons use loop-shared projections throughout.

Table 7: Auxiliary capacity channels at L=2, backbone 134.5M. Streamed lists nominal active weight parameters (embedding/head counted once); Gathered lists private token-table parameters separately. Shared S-PLE projections are included in streamed weights. The fused adapter adds no separate op but widens the GEMM; adding it *on top of S-PLE* increases measured decode TPOT by 4–18% across scales, and the 2×2 ablation isolates each channel's increment (Appendix K). The full stack has 0.58× the dense reference's nominal streamed weights (139.7M vs. 241M).

Channel	Aux.	Streamed	Gathered	Primitive
plain loop	0	134.5M	0	—
`zf8`	2.95M	137.5M	0	wider GEMM
`sp32`	95.5M	135.6M	94.4M	gather + projections
`sp64`	191.0M	136.7M	188.7M	gather + projections
both / preinit	194.0M	139.7M	188.7M	both primitives
dense 240M	—	241M	0	unique layers

Table 8: Arm ledger for the 135M fleet (Table 5) plus the distilled-cooldown pair. The official hf-135m row of Table 5 is an external checkpoint, not fleet-trained; the distilled-cooldown twins are 20B-token continuations of the preinit arm, reaching 45B tokens in total (Appendix F.1), not part of the 100B-token census.

Arm	Tokens	Role
base-135m	100B	dense baseline
base-135m-ple64	100B	dense + S-PLE64 (input-space control)
base-240m	100B	dense 2×-depth reference
loop-l2 / loop-l3	100B	plain recurrent baselines
loopit-l2	100B	PLT substrate, no channels
loopit-l2-`sp32`/`sp64`/`sp128`	100B	input-space width sweep
loopit-l2-`zf8`	100B	weight-space channel alone
loopit-l2-`zf8-sp64`	100B	both channels
loopit-l2-`zf8-sp64`-pi	100B	teacher-preinitialization condition
loopit-l3	100B	three-loop variant
unified-codebase replicates	100B / 50B	two-seed replication set, 7 runs (Table 9)
cooldown CE / KD twins	45B	distilled-cooldown pair (Appendix F.1)
hf-135m (official)	—	external reference, not trained here

D Seed replication under the unified codebase

The two central channel arms of Table 5 (S-PLE64, zFLoRA8), two 135M dense controls (dense-zf16, dense-135m), and the 906M A3 arm were retrained from scratch with independent seeds under a single unified code state (100B tokens at 135M; 50B at 906M; two seeds for each 135M channel arm, one run for every other entry of Table 9). WikiText-2 token-level validation perplexity (266k held-out tokens, seq. 2048) agrees across seeds to <0.25%: S-PLE64 18.41 vs. 18.38; zFLoRA8 19.59 vs. 19.55. At 906M, the unified-codebase A3 replicate (13.55) sits 1.2% below the original-recipe run (13.71); recipe and seed remain confounded at this scale, so this figure bounds their combined variation rather than isolating seed noise. The likelihood ordering matches Table 5: S-PLE64 < zFLoRA8 < dense; the added dense-zf16 control slots between zFLoRA8 and dense. The replication supports the *ordering*, not the headline magnitude: the S-PLE64−zFLoRA8 gm-ppl gap is 1.82 (seed 1) and 2.47 (seed 2) under the unified recipe, versus 3.18 in the original-recipe Table 5 — a 22–43% smaller effect under a confounded recipe change (dropout removed), so magnitudes should be read as single-recipe, single-seed. No replicate of the no-channel baseline (loopit-l2) exists in either recipe.

Table 9 reports harness geometric-mean perplexity (LAMBADA×WikiText, as in Table 5) for every completed replicate. Absolute values shift against the original runs: the original fleet trained with dropout 0.1 on attention and loop paths, which the unified codebase omits. The shift is +0.85 gm-ppl for the loop-free dense control (attn-dropout only), +0.88 for

zFLoRA8, and +2.24 for S-PLE64; within-generation seed agreement is 0.06 (S-PLE64) and 0.71 (zFLoRA8) gm-ppl points. We therefore treat cross-generation absolute comparisons as recipe-confounded and rest seed-variance claims on within-generation pairs; avg-7 reproduces across seeds within ≈1 point.

Table 9: Seed replication under the unified codebase (100B tokens, 135M unless noted). val = WikiText-2 token-level validation perplexity; gm = harness gm-ppl as in Table 5; paper-arm gm from Table 5 (original recipe, single seed).

Arm	val s1	val s2	gm s1	gm s2	paper-arm gm
S-PLE64	18.41	18.38	25.15	25.21	22.91
zFLoRA8	19.59	19.55	26.97	27.68	26.09
dense-zf16	—	20.43	—	29.19	—
dense-135m	—	20.81	—	30.40	29.55
A3-840M (50B)	—	13.55	—	15.03	—

E Probe measurements: loop specialization and capacity sweeps

This appendix expands the mechanism analysis of Section 5 with the full measurements. Evidence sources: four probes on the saved 100B-token `ckpt_final.pt` checkpoints (plain looped-L2 (no PLT machinery), base PLT without channels, PLT with S-PLE (sp64), `zf8`, `zf8`+sp64, and the `zf8`+sp64 sibling pre-initialized from the 2T-token SmolLM2-135M teacher (preinit)) — CKA on the full wikitext2 validation split (130 sequences × 2048 tokens; single seed; uncentered linear CKA between loop-1 and loop-2 token representations plus mean per-token cosine), ablations on 8 sequences × 2048 tokens (rerun on the full validation split with bootstrap CIs for the two surviving checkpoints; Table 12) — plus two matched-budget capacity sweeps read from in-training validation trajectories: adapter rank/variant at 25B tokens (`zf8`/`zf16`/`zf32`/lora-r8, all +shared-S-PLE, identical recipe; r16 died at 18B and joins at 15B) and S-PLE width at 50B tokens (ple_dim 32/64/128, no adapters; `ckpt_50.00B.pt` verified in all three runs).

Representation-similarity metric. We use the uncentered variant of linear centered kernel alignment (CKA; Kornblith et al., 2019). For token-representation matrices X and Y from the two loops, the score is

$$\mathrm{CKA}_{\mathrm{uncentered}}(X, Y) = \frac{\|X^\top Y\|_F^2}{\|X^\top X\|_F \, \|Y^\top Y\|_F}.$$

We do not subtract the token-wise feature means. Higher values indicate greater representational similarity; the score does not identify a computational role.

Table 10: Loop divergence (loop-1 vs loop-2) at 100B: mean/min/max per-layer uncentered linear CKA over the 30 layers, and the final-layer CKA and mean per-token cosine of the post-norm representations the LM head reads. Lower CKA/cos indicates less similar representations, not a distinct computational role.

Model	mean CKA	min CKA	max CKA	final CKA	final cos
looped-L2 (plain)	0.675	0.296	0.961	0.626	0.349
PLT-L2 (no channels)	0.506	0.286	0.720	0.617	0.240
LooPiT-L2 + S-PLE	0.509	0.172	0.732	0.643	0.246
PLT-L2 + `zf8`	0.479	0.179	0.787	0.560	0.275
LooPiT-L2 + `zf8`+S-PLE	0.497	0.136	0.836	0.533	0.275
`zf8`+S-PLE + preinit	0.690	0.344	0.968	0.548	0.297

Takeaway 1 — input tables help more at the tested sizes. At the observed 25B checkpoint, channel-free PLT has validation NLL 3.1561, weight-only zFLoRA8 3.1473, and input-only S-PLE64 3.0652. The respective improvements are 0.0088 and 0.0910; auxiliary

Table 11: Causal S-PLE channel ablation at 100B: next-token CE change (Δnll) when zeroing or fixed-vocab-shuffling each loop's tables at inference (8 val sequences × 2048 tokens). The scratch arms' weights were purged, so these rows remain 8-sequence-subset estimates; Table 12 reruns the two surviving checkpoints on the full validation split. base ppl is the probe harness's own subset scoring and is not comparable to trainer-parity validation perplexity (Table 12); the Δnll columns are the measurement of record.

Model	base ppl	zero L1	zero L2	shuf. L1	shuf. L2
LooPiT-L2 + S-PLE	12.82	+7.48	+1.90	+6.08	+1.83
LooPiT-L2 + `zf8`+S-PLE	12.68	+7.77	+1.86	+5.79	+2.02
`zf8`+S-PLE + preinit	12.27	+0.61	+0.35	+1.02	+0.61

Table 12: Full-validation causal rerun on the two surviving checkpoints (130 val sequences × 2048 tokens; Δnll mean ± bootstrap 95% CI half-width over sequences, 2,000 resamples; shuffle = 5 seeds). Trainer-parity base ppl: 17.11 (preinit; run-log 17.106) and 13.67 (A3 840m; run-log 13.71). The preinit deltas preserve the qualitative ordering of the 8-sequence estimates in Table 11.

Model	zero L1	zero L2	shuf. L1	shuf. L2
`zf8`+S-PLE + preinit	+0.602±0.016	+0.390±0.013	+1.012±0.031	+0.644±0.018
A3 `zf8`+`sp128` (840m)	+0.768±0.013	+0.265±0.011	+0.879±0.019	+0.344±0.013

Table 13: Adapter geometry at 100B (pure weight inspection, no forwards): cross-loop principal-subspace overlap of the per-loop compressors (C_j, QKV/gate/up) and expanders (E_j, o/down), effective rank of the stacked $[C_0|C_1]$ (16 columns), second-loop/first-loop compressor norm ratio ($\|C_1\|/\|C_0\|$, so > 1 means the second loop's compressor is larger), and the parameter census (tied embeddings counted once; totals include the 2.2M loop-shared S-PLE projections).

Model	comp ov.	exp ov.	eff. rank	$\|C_1\|/\|C_0\|$	PLE tbl	adapters	total
`zf8`	0.240	0.185	14.33/16	1.46	0 M	2.95 M	137.5 M
`zf8`+S-PLE	0.215	0.190	14.47/16	1.49	188.7 M	2.95 M	328.5 M
`zf8`+S-PLE + preinit	0.186	0.166	14.92/16	1.36	188.7 M	2.95 M	328.5 M

Table 14: Causal adapter-path ablation: next-token CE change (Δnll) when zeroing each loop's adapter weights (for *zero*, the weightless AvgPool-injection/merge scaffold of the ablated pass is neutralized as well), scrambling their output-channel wiring (spectrum-preserving row permutation), or exchanging the two loops' adapters (*swap*). S-PLE tables intact throughout; 8 val sequences × 2048 tokens; base ppl as in Table 11 (probe-harness subset scoring, not trainer-parity). Top: 100B `ckpt_final` checkpoints. Bottom: matched 25B rank/variant suite (zf-r8/zf-r32 = `zf` at rank 8/32, lora-r8 = vanilla LoRA control; identical recipe, all +shared-S-PLE).

Model	base ppl	zero L1	zero L2	shuf. L1	shuf. L2	swap
`zf8`	13.90	+6.19	+3.53	+7.28	+5.84	+11.26
`zf8`+S-PLE	12.68	+2.00	+0.24	+2.63	+0.52	+7.87
`zf8`+S-PLE + preinit	12.27	+0.39	+0.43	+0.86	+1.04	+6.79
`zf8`@25B	15.10	+1.16	+0.11	+3.64	+0.26	+6.14
`zf32`@25B	14.80	+2.77	+0.22	+7.73	+0.50	+5.58
lora-r8@25B	14.89	+2.00	+0.19	+3.38	+0.32	+11.21

parameter counts differ by approximately 65×. Adding zFLoRA8 or zFLoRA32 to S-PLE64 gives NLL 3.0433 or 3.0375, respectively. These comparisons concern the tested budgets, not equal auxiliary capacity. The independent weight-only rank sweep is reported in Appendix J.1.

Our interpretation is conditioning rather than additional information: the trunk already receives token identity through the shared embedding. S-PLE supplies a learned vector for each (layer, loop, token), while adapters modify transformations of the hidden state. Large zeroing and shuffling costs (Tables 11 and 14) show that the trained networks use the channels; they do not identify why input tables yield the larger training-time gain.

Takeaway 2 — within-channel size responses differ. Conditional on S-PLE64, rank 16 is 0.0013 nll worse than rank 8 at 15B tokens, whereas rank 32 is 0.0058 better at 25B. Vanilla LoRA-r8 is 0.0090 ahead of zFLoRA8 at 25B (3.0343 vs. 3.0433). These are single-seed point estimates, not saturation or parameterization-efficiency results. We match the evaluation nearest each nominal token budget within 0.01B tokens; stopped runs contribute no later points. In the separate weight-only campaign, rank 32 gains 0.0089 nll at 50B without closing the gap to S-PLE-only references (Appendix J.1). For S-PLE widths 32/64/128, the 50B NLLs are 3.0089/3.0022/3.0506, versus 3.0980 without S-PLE. At 100B they are 2.8869/2.8774/2.9028. Width 64 has the best point estimate at both budgets; increasing table capacity is not monotonically beneficial in this sweep. Rare-token estimation is a possible explanation, but token-frequency gradient coverage was not measured.

Takeaway 3 — ablation reliance is not marginal value. Ablation sensitivity need not track validation-loss gains from added capacity. At 25B, zeroing loop-1 adapters costs +1.2 nll for `zf8` and +2.8 for `zf32`, although their validation nll differs by 0.0058. Across S-PLE widths 32/64/128, the loop-1 zeroing costs are +3.0/+5.9/+4.3 nll, a non-monotone pattern. These interventions measure dependence of each *trained* network on its channel; they do not estimate how much a larger channel improved training, nor show that the extra parameters learned only redundant features.

Takeaway 4 — loop 1 is more ablation-sensitive. For S-PLE (Table 11), zeroing loop-1 tables costs +7.5/+7.8 nll in two scratch arms, versus about +1.9 for loop 2. For adapters (Table 14), the both-channel scratch arm costs +2.0 when loop 1 is zeroed versus +0.24 for loop 2; loop-2 adapter zeroing costs +0.1–+0.5 across the S-PLE-bearing 25B rank suite. Without S-PLE, the `zf8`-only arm is more sensitive to its loop-2 adapter (+3.5) than the both-channel arm is (+0.24), consistent with some substitution between channels, but these ablations do not uniquely assign a computational role to either one. The loop-2 compressor is nonetheless 1.36–1.49× larger in norm than loop 1's (Table 13), so norm does not track this ablation pattern. At 906M, A3's full-validation zeroing costs are +0.768 (loop 1) and +0.265 (loop 2), with the same ordering under shuffling (Table 12).

Takeaway 5 — adapter assignment by loop matters. In Table 14, swapping the two loops' adapters costs +5.6–+11.3 nll, more than zeroing either one, including under preinit (swap +6.8 versus zero +0.4). This establishes sensitivity to the trained loop assignment; a swap is an out-of-distribution intervention and does not by itself quantify the adapters' training-time value. Shuffling output-channel wiring also costs more than zeroing loop 1, consistent with structured corrections rather than signal presence alone. Cross-loop subspace overlap is 0.17–0.24 and the stacked compressor's effective rank is 14.3–14.9 of 16 (Table 13); these are descriptive geometry measurements, not an independent causal test of the source of the gain.

Takeaway 6 — divergence differs from plain looping. The plain looped baseline has higher loop-1/loop-2 mean CKA (0.68) than the scratch PLT-family variants (0.48–0.51, Table 10). PLT changes right-shift and shared-cache structure together, so this comparison does not attribute the CKA difference to either component. The original prediction that S-PLE would increase divergence relative to `zf8`-only is not confirmed. The single-seed width readings at 50B (0.544/0.508/0.582 for 32/64/128) likewise show no monotone relationship with CKA.

Takeaway 7 — preinit increases alignment and reduces ablation costs. Under preinit, ablating either S-PLE table costs +0.35–1.0 nll versus +1.9–7.8 in the scratch runs (Table 11); mean loop-1/loop-2 CKA is higher (0.69 vs. 0.48–0.51, Table 10). These obser-

vations are compatible with reduced reliance on S-PLE after trunk preinitialization, but do not distinguish that account from other effects of initialization and training. Preinit improves validation ppl by 0.37 over its scratch sibling (17.11 vs. 17.48) and passes the dense-240M reference (17.35; Table 15). Its adapter zeroing costs are also lower (+0.4), while swapping loop assignments still costs +6.8 (Table 14): lower individual ablation cost does not imply interchangeability.

Table 15: Wikitext validation-perplexity anchors for the 135M-class prose claims of Appendices B.3, F.1 and J (trainer-parity val ppl; the preinit, CE-twin, and KD-twin values are printed in Tables 16 and 20; the dense-240m reference and the scratch sibling were previously prose-only).

Run	val ppl
base-240m (dense 2× depth, 100B)	17.35
LooPiT-L2 `zf8`+`sp64` (scratch sibling, 100B)	17.48
LooPiT-L2 `zf8`+`sp64` (preinit, 100B)	17.11
CE cooldown twin (45B)	17.09
KD cooldown twin (45B)	15.90
KD cooldown twin (1.7B teacher, 45B)	15.80

F Knowledge-distilled cooldown: direction, dose, and maturity

F.1 Cooldown comparison and training objective

Both twins resume the pre-init arm from its 25B-token checkpoint and anneal the LR linearly to zero over a 20B-token tail; the control trains plain CE, the KD twin adds gated forward-KL output distillation (λ=0.5, T=2, final-loop logits, gated to tokens where the teacher's hard-label NLL beats the student's) from the official SmolLM2-360M (2.7× the backbone). Distillation inside pretraining is established (Gemma Team, 2024; Busbridge et al., 2025; Muralidharan et al., 2024); the observation here is that the distilled anneal tail surpasses the 100B CE parent in this single-seed comparison: the CE twin reaches 17.09 val ppl at 45B tokens (within 0.03 of the parent's 17.11), and the KD twin reaches 15.90 (−1.19 ppl; gm-ppl 20.52 vs. 21.51), a 1.7B-teacher twin 15.80 (Table 16). Average accuracy is preserved (avg-7 55.9 vs. 55.6) with a BoolQ gain (+2.3) but regressions on the two reported generative metrics (LAMBADA −1.8, CoQA F1 −2.4; Table 17): the gated KL improves likelihood without guaranteeing better generated answers. Forward KL beats reverse KL at every dose where both were measured (λ=0.25, both maturities). The twins are single-seed. The 45B-versus-100B comparison concerns student training tokens; it does not account for teacher forward computation or the initialization checkpoint's prior training.

F.2 Dose, maturity, and downstream outcomes

Three observations (Table 16).**(i) An early anneal matches the full schedule.** The CE twin reaches 17.09 at 45B tokens, within 0.03 ppl of the parent's 100B final (17.11): the extra 55B tokens do not improve this validation-ppl point estimate under these schedules. **(ii) Distillation improves this anneal's likelihood.** The KD twin lands at 15.90 (−1.19 ppl vs. the CE twin) — well past the dense-240M baseline (17.35) — beats the 100B parent's gm-ppl headline (20.52 vs 21.51). **(iii) The likelihood gain preserves average downstream accuracy.** avg 52.5 vs. 52.3, avg-7 55.9 vs. 55.6; the largest reported gain is on BoolQ (60.6 vs. 58.3, Table 17; remaining per-task columns ship with the released evaluation pack). However, the consistent regressions are LAMBADA accuracy (44.3 vs. 46.1; Table 17) and CoQA F1 (41.7 vs. 44.1). The 360M teacher's native CoQA F1 (58.0) does not transfer through this token-level KL recipe; these single-seed results do not isolate whether the objective, teacher, schedule, or their interaction causes the regressions. Perplexity gains therefore do not translate uniformly to generative robustness.

A further cooldown uses the official SmolLM2-1.7B teacher (12.6× the backbone), while also changing the distillation weight to λ=0.25 and truncating to the top 1024 logits. It reaches

15.80 validation ppl and 19.50 gm ppl, versus 15.90 and 20.52 for the 360M-teacher twin. Teacher size, dose, and truncation are confounded, so this is not a teacher-scaling or saturation test. All twins are single-seed.

Table 18 reports 2B-token anneals from scratch and from the 25B preinitialized checkpoint. At the common dose λ=0.25, forward KL beats reverse KL by 0.78 ppl from scratch and 0.18 ppl after pretraining. The reverse-KL dose comparisons use different intervals: raising λ from 0.25 to 0.5 worsens scratch ppl by 1.10, while lowering it from 0.25 to 0.1 worsens mature ppl by 0.38. These observations do not identify a common dose-response curve. Mature forward-KL results differ by 0.02 ppl between doses 0.25 and 0.5. All tested mature KD variants reach or improve on the 20B CE-tail endpoint using a 2B tail, but the table has no matched 2B CE control: it cannot separate distillation from the shorter annealing schedule.

Table 16: Cooldown A/B on the pre-init arm (Appendix F.1): resume at 25B, anneal 20B tokens (LR $3\times10^{-3} \to 0$), evaluate the 45B finals. val = token perplexity (WikiText-2 validation); accuracy columns follow Table 5's conventions (strategy-A decoder); gm ppl = geometric mean of LAMBADA and WikiText perplexity.

Arm	tokens	val ppl↓	gm ppl↓	avg	avg-7	CoQA F1
pre-init parent (100B schedule)	100B	17.11	21.51	51.5	54.5	**44.2**
CE cooldown twin	45B	17.09	21.40	52.3	55.6	44.1
KD cooldown twin (SmolLM2-360M)	45B	15.90	20.52	**52.5**	**55.9**	41.7
KD (1.7B teacher)	45B	**15.80**	**19.50**	52.4	55.7	42.4

Table 17: Per-task columns for the cooldown twins (0-shot, strategy-A) beyond Table 16: the KD twin's BoolQ accuracy gain and LAMBADA accuracy regression, backing the prose of Appendix F.1. CoQA F1 is repeated from Table 16; the 360M teacher's native CoQA F1 is 58.0.

Arm	BoolQ	LAMBADA	CoQA F1
CE cooldown twin	58.3	46.1	44.1
KD cooldown twin	**60.6**	44.3	41.7

Table 18: KD direction and dose at two maturities: final validation ppl of 2B-token anneals (LR $3\times10^{-3} \to 0$, T=2, gated, full vocabulary, 360M teacher, 8-GPU probes at 524,288 tok/step). Scratch = 2B from-scratch run; mature = resumed from the 25B pre-init checkpoint (25→27B). Lower is better; dashes = not measured.

direction	λ	scratch 2B	mature 25→27B
fwd	0.25	26.681	**16.494**
fwd	0.5	—	16.512
rev	0.1	—	17.059
rev	0.25	27.460	16.676
rev	0.5	28.559	—

G Quantization robustness: full simulated policy

Quantization robustness (simulated). A calibration-free RTN stress test uses int4 group-128 weights and dynamic int8 activations (Appendix G). At 906M/50B, dense T1 degrades by 1.68 ppl versus 1.05 for the both-channel A3 arm. Lower marginal weight error does not explain this difference: RTN weight error is equal or higher for the looped arms. A 135M grid crosses RTN with calibrated GPTQ (Frantar et al., 2023). Under RTN w4a8, the preinitialized both-channel stack degrades less than the adapter-budget-matched dense-zf16 control (3.89 vs. 6.00 ppl; degradation gap 2.11, 95% CI $[1.93, 2.29]$). The gap falls to 0.11 $[-0.01, 0.23]$ for a scratch-trained looped-zf8 control, but this control also removes S-PLE

and PLT machinery: it does not isolate initialization. Under GPTQ the advantage reverses ($-0.30/-0.46$ at w4a8/w4a16); role-conditioned calibration reduces the difference until its CI includes zero, without establishing equivalence. Robustness therefore depends on the stack and quantizer. These simulations differ from the handset's calibrated AIMET per-output-channel w4a16-kv8 path; neither validates its quality. AWQ (Lin et al., 2024) is not evaluated.

Table 19 reports the full simulated policy: wikitext-2 validation perplexity (266K tokens, the training eval path) under a uniform calibration-free round-to-nearest policy — weights int4 symmetric group-128, activations dynamic per-token int8, no calibration; norms and biases full precision. The 906M pair shares one recipe and token budget (Table 6); the fp column re-evaluates the released checkpoints (within 0.05 of the 50B-grid values in Table 6); the T2 row applies the same policy to the dense 1.7B scale reference (same corpus, 50B tokens). Activations to int8 cost at most 0.22 beyond w4a16 among the fleet arms (0.19/0.13 for the looped arms, 0.22 for the dense twin): quantization fragility appears concentrated in the backbone/adapter weight matrices rather than in the input-space channel. These quality measurements do not establish which resource limits decode latency. Per-weight RTN error under the same policy is equal or higher for the looped arms (matched pair: 0.118–0.124 vs 0.118 relative error; kurtosis 5.3 vs 3.6), so lower marginal weight error alone does not explain the widening. LoopQ (Fang et al., 2026) reports looped backbones as more PTQ-fragile than dense ones; our looped arms additionally carry the input-space channel — a candidate source of the discrepancy, though the regimes also differ (calibrated PTQ there, uncalibrated RTN here). The calibrated-weight-quantizer sensitivity check below partially reconciles the two: under GPTQ the budget-matched advantage does reverse at 135m, while the advantage over the plain-dense reference persists.

Table 19: Simulated weight-4-bit robustness: wikitext-2 validation perplexity (266K tokens, the training eval path) under the policy described in the text. The 906M pair shares one recipe and token budget (Table 6); the T2 row applies the same policy to the dense 1.7B scale reference (same corpus, 50B tokens). The two 135M rows are re-measured under the hardened per-block protocol of the follow-up grid (Table 20); values agree with the original run to $\leq$0.03.

Model	fp	w4a16	w4a8	w4a8 (tables fp)
SmolLM2-135M (stock reference)	16.37	33.89	35.11	—
LooPiT-L2 `zf8`+`sp64` (preinit) 135M (100B)	17.11	20.81	21.00	20.97
T1 base-840m (dense 12L)	14.62	16.08	16.30	—
A3 `zf8`+`sp128`	13.67	14.59	14.72	14.70
T2 base-1p7b (dense 24L)	12.69	13.65	13.85	—

Hardened pair-2 grid: quantizer $\times$ activation precision. A follow-up run hardens the 135m measurement (the pre-initialized looped row confounds architecture with initialization; the scratch `zf8` control below additionally removes S-PLE and PLT machinery, so it does not isolate these effects) and crosses the policy above with a calibration-based weight quantizer. Hardening: per-block NLLs (130 blocks, the training eval path) with paired-bootstrap 95% CIs (10k draws, one block-index matrix shared by every config, model, and the cross-model gap statistic), a strategy-A decoder, and first/last-fp decompositions (adapter-weights-fp, head-fp, PLE-tables-fp). Quantizer: stage-wise *sequential* GPTQ (Frantar et al., 2023) on the same per-(out, group-128) symmetric format — Hessians $X^\top X$ accumulated at every linear call site from wikitext-2 *train* tokens (the validation stream is the eval; calibrating on it would leak), loop iterations accumulated into the shared block-weight Hessian (the property stock GPTQ tooling cannot see), blocked error feedback, and weights re-solved after every layer stage on the partially-quantized model. The GPTQ weight preset is built once per model and shared by the w4a8 and w4a16 columns; embedding-path tables (PLE LUTs, tied embeddings) keep per-row RTN. Table 20 reports the grid.

Three comparisons qualify the robustness result. *(i) RTN favors the tested preinitialized stack.* The degradation-gap statistic (dense-zf16 Δ minus looped Δ; positive means less degradation for looped) is $+2.12$ $[1.94, 2.29]$ at w4a16 and $+2.11$ $[1.93, 2.29]$ at w4a8. For

Table 20: The 135m grid: wikitext-2 validation ppl (266K tokens) under {RTN, GPTQ} × {w4a16, w4a8}; GPTQ = sequential GPTQ weights (shared per-model preset). Parenthesized values are Δ = ppl − fp (computed from unrounded values, so they can differ by 0.01 from displayed-column subtraction); paired-bootstrap 95% CIs for the key comparisons are in the text. fp anchors reproduce the training run logs (preinit $|\Delta| = 0.005$; scratch $|\Delta| = 0.003$); trio rows agree with the original fleet run to ≤0.03; GPTQ was not re-run for the scratch row (—).

Model	fp	w4a16	w4a16 GPTQ	w4a8	w4a8 GPTQ
SmolLM2-135M (stock reference)	16.37	33.89 (+17.51)	28.90 (+12.53)	35.11 (+18.74)	30.10 (+13.73)
dense-zf16-135m (adapter-matched ctrl)	20.19	26.01 (+5.81)	24.17 (+3.98)	26.19 (+6.00)	24.63 (+4.43)
LooPiT-L2 `zf8`+`sp64` (preinit) 135M	17.11	20.81 (+3.70)	21.54 (+4.44)	21.00 (+3.89)	21.84 (+4.73)
looped-`zf8`-135m (scratch, machinery off)	19.59	25.07 (+5.48)	—	25.48 (+5.89)	—

scratch looped-zf8 without S-PLE or PLT machinery, the gaps are +0.34 [0.22, 0.45] and +0.11 [−0.01, 0.23]. The control matches adapter parameters (2.95M), not the preinitialized stack's additional 191M S-PLE parameters. These results do not attribute the difference to initialization alone.

(ii) GPTQ reverses the degradation ordering. With GPTQ, the preinitialized stack's degradation gap against dense-zf16 is −0.46 [−0.55, −0.37] at w4a16 and −0.30 [−0.41, −0.20] at w4a8. Three additional Gauss–Seidel sweeps, remeasuring Hessians on the quantized model, do not improve this solution; this is a stability check, not a guarantee of a global optimum. GPTQ reduces local reconstruction error at all 721 linear call sites, yet gives 0.7–0.8 worse perplexity than RTN for this looped stack. Thus local reconstruction error does not predict end-to-end perplexity in this comparison.

(iii) The comparison depends on the reference and policy. The preinitialized stack degrades less than stock SmolLM2 under every tested policy. Allowing each model its best tested quantizer, preinit+RTN has a smaller degradation than stock+GPTQ by 8.8 [8.2, 9.4] at w4a16 and 9.8 [9.2, 10.5] at w4a8, and than dense-zf16+GPTQ by 0.27 [0.18, 0.37] and 0.54 [0.44, 0.65]. Under RTN w4a8, keeping adapter weights floating point recovers 0.02 ppl for the looped stack versus 1.07 for dense-zf16; keeping the head floating point recovers 0.61 (untied export head) versus 1.48 (tied control). Keeping the stock model's embeddings and head floating point recovers 8.6 of its 18.7 ppl degradation. These decompositions are specific to the reported export and quantization policies.

The grid uses one quantized copy per shared weight. GPTQ accumulates a single Hessian across loop roles; calibration uses WikiText-2 training data and evaluation uses its validation split, so cross-corpus calibration robustness is untested. A role-conditioned variant instead uses one Hessian and one quantized copy per (weight, loop role), covering 991 role pairs over 721 call sites. The dense control is single-role and unchanged. The w4a8 degradation gap moves from −0.30 [−0.41, −0.20] to +0.07 [−0.03, 0.18]: its interval includes zero, but this does not demonstrate equivalence or restore RTN's 2.11 advantage. Role-specific copies also increase weight storage, so this is not a measurement of the shared-weight deployment graph. LoopQ's other mechanisms, including role-specific activation scaling and cross-loop state alignment, remain untested here. Finally, the handset uses calibrated AIMET per-output-channel w4a16-kv8, not this symmetric group-128 RTN/GPTQ simulation.

H Extended evaluation at the 906M-backbone finals

Because 0-shot ARC-c and Winogrande sit near chance at this scale, Table 21 reports selected held-out measurements for all seven arms: web perplexity with paired-bootstrap 95% CIs, code perplexity, 5-shot ARC-c, CoQA F1, and CNN/DailyMail (Hermann et al., 2015) ROUGE (Lin, 2004) (strategy-A decoding for generation). On web, T1's interval is disjoint from every looped arm's; A3 and T3 have overlapping intervals, which do not establish equivalence. The displayed code point estimates follow the validation ordering except for the A2–L6 tie. The table does not report uncertainty for code, accuracy, or rouge. Generative quality does not track likelihood: the weight-only arm A2 has the highest measured rouge-1 (23.6) despite

ranking fifth on validation ppl, with A1 close behind (23.4). CoQA F1 ranges from 45.3 to 49.3 among the 906M-class arms; the depth-doubled dense reference reaches 52.5.

Table 21: Selected extended-eval results at the 50B finals (all arms, strategy-A decoder for generation). Web ppl includes paired-bootstrap 95% CIs; other columns are point estimates. ARC-c^{5s} = 5-shot acc-norm; rouge-1 at 500 CNN/DM test docs. T2 (1.71B) is the dense depth reference. Bold = column best.

Arm	web ppl [CI]	code ppl	ARC-c^{5s}	CoQA F1	rouge-1
T1 dense-906M	18.17 [18.01, 18.34]	4.75	33.3	45.3	18.1
T2 dense-1.71B	**16.14** [16.00, 16.29]	**4.29**	**39.1**	**52.5**	19.3
L6 looped	17.55 [17.40, 17.72]	4.63	33.0	48.6	18.0
A2 zf8 (weight-only)	17.60 [17.45, 17.77]	4.63	33.9	47.7	**23.6**
A1 sple64	17.40 [17.25, 17.56]	4.57	34.3	47.1	23.4
T3 zf8+sple64	17.36 [17.21, 17.53]	4.56	34.2	48.6	20.7
A3 zf8+sple128	17.27 [17.11, 17.43]	4.54	35.6	49.3	19.6

I Generative evaluation is strategy-sensitive

CoQA F1 for PLE models collapses from ≈40 (strategy A, train-consistent refinement prefill) to ≈0.5 (strategy B, single-pass prefill), demonstrating sensitivity to the prefill implementation. All reported task results use strategy A. In the 135M H100 bf16 implementation, median time to first token (TTFT) is 72.0 ms for PLT at L=2 and 38.4 ms for dense (batch 1, 4096-token prompt, 128 generated tokens, 10 iterations). The strategy-specific work occurs during prefill; both strategies use the same subsequent decode path. These GPU timings are separate from the NPU measurements in Appendix K. Alternative strategies are diagnostic only: strategy B's generation is degraded fleet-wide (CoQA F1 0.5–0.9 points at 135M) and the two-step strategy-C decoder is not train-matched. The 906M-class generative scores in Section 4.3 and Table 21 additionally required routing generation through the model-native train-parity decoder: lm-eval's batched generate path degrades LooPiT generation (CoQA F1 0.04 vs. 0.47 on the 0–1 scale for the same checkpoint).

J Training trajectories and capacity sweeps

Training trajectories are regenerated from per-run `metrics.jsonl` files. Endpoint and latency panels use the reported evaluation tables and benchmark logs; the plotting scripts record those inputs. Figures 3 and 4 show token-matched validation trajectories for the 906M and 135M studies, respectively.

Token-matched trajectories (Figure 4). On the token axis the channel ordering is established at the first eval and stable through the schedule: input-space specialization separates from plain looping immediately and never re-converges (S-PLE64 stays ≥0.07 loss ahead at every point), while the weight-space-only arm tracks plain looping within ≈0.09 loss throughout. Pre-initialization leads every from-scratch arm at the first evaluation (0.5B tokens). During annealing, the dense 2×-depth reference and the scratch combined-channel model briefly outperform it; preinit finishes best among the plotted 100B census arms. Figure 5 replots the fleet against a backbone-compute proxy.

Backbone-proxy endpoints (Figure 5). The channel ordering persists when token budgets are rescaled by executed backbone size. At the common proxy endpoint, preinit has 17.11 validation ppl versus 17.35 for dense-240M. All primary L=2 scratch arms share the same token budget and backbone execution count, but channel-specific work is omitted. The preinitialized arm additionally inherits the teacher's prior training. This view therefore does not establish an exact compute-efficiency advantage.

Conditional rank and width trajectories (Figure 6). The older rank trajectories measure higher adapter rank *conditional on S-PLE* at the last common grid point (18B

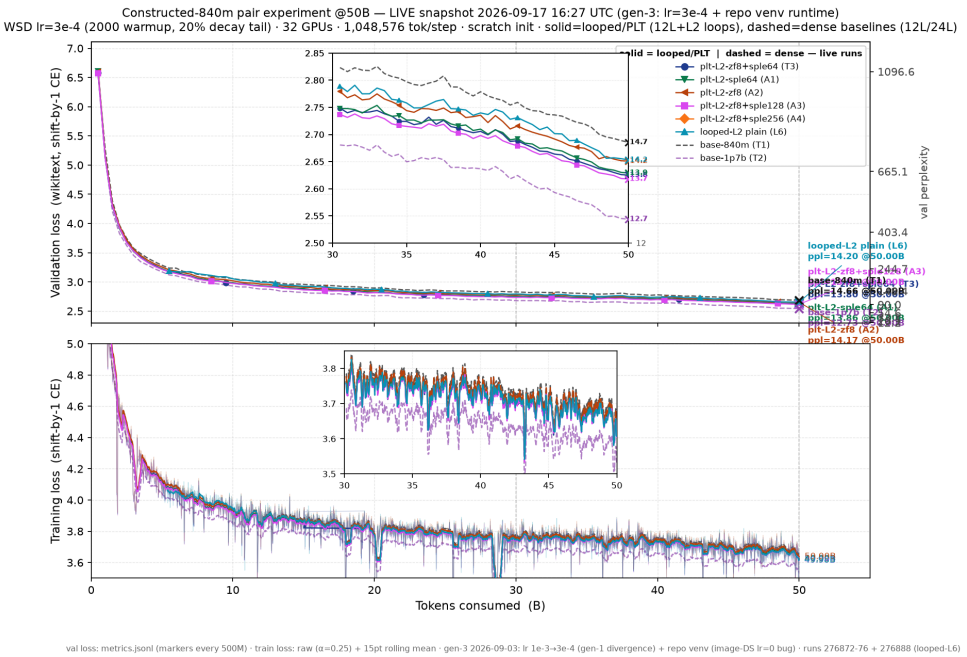


Figure 3: 906M-backbone study, validation loss through 50B tokens (inset: 30–50B tail). Solid = plain-looped/PLT-family arms (12L trunk, L=2); dashed = dense baselines (12L, 24L).

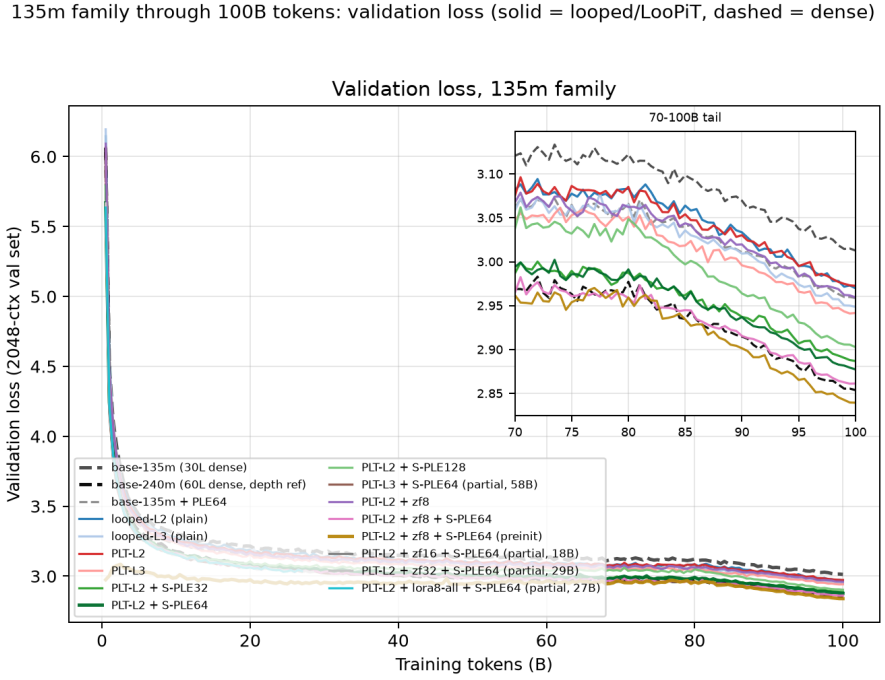


Figure 4: 135m family, validation loss through 100B tokens (inset: 70–100B tail). Solid = plain-looped/PLT-family arms; dashed = dense baselines (30L base-135m, 60L base-240m depth reference). Token-matched companion to Figure 5's compute-matched view (same canonical run dirs and arm colors); partial campaigns (zFLoRA rank sweep, lora8, L3+S-PLE64) are drawn over their completed ranges only.

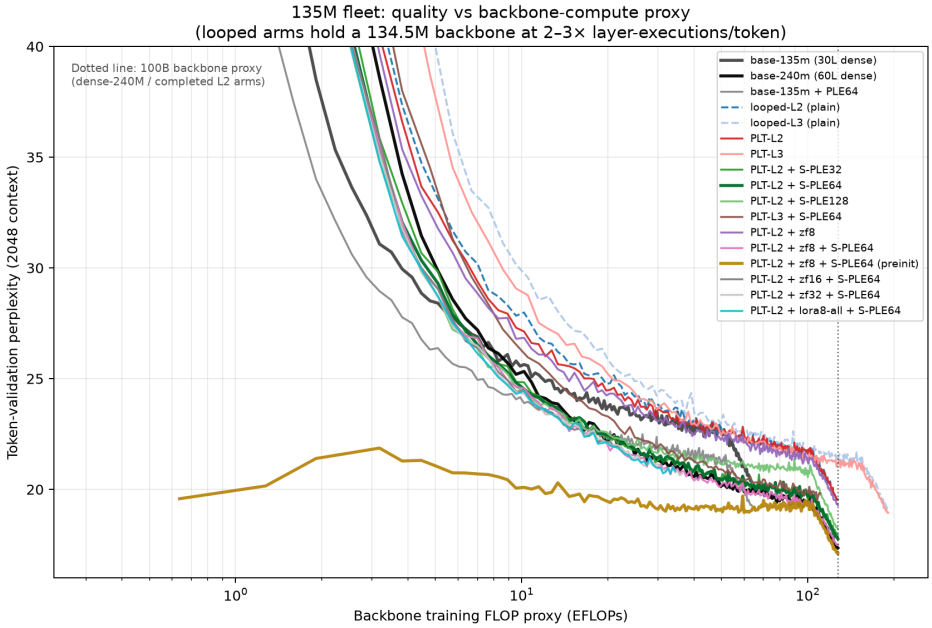


Figure 5: 135M fleet: validation perplexity against a backbone training FLOP proxy, $6N_{\text{nonembed}}L_{\text{exec}}$ per token. The proxy counts executed backbone parameters but excludes attention matrix products, output-head work, auxiliary channels, and prior teacher training. Completed L=2 runs and depth-doubled dense share the dotted endpoint under this approximation, not exact compute accounting. The vertical axis is restricted to 16–40 ppl.

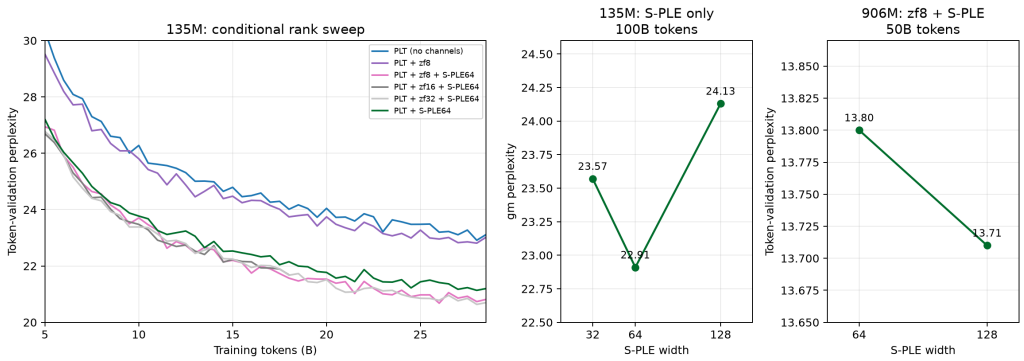


Figure 6: Conditional rank and width sweeps. Left: observed validation perplexity for ranks 8/16/32 with S-PLE64, plus single-channel and channel-free references; r16/r32 end at 18B/28.5B. Center: 135M S-PLE-only width sweep, gm ppl at 100B. Right: 906M both-channel width sweep with rank 8 fixed, token validation ppl at 50B. The width panels use different metrics and are not a cross-scale curve.

tokens): ranks 16/32 reach 21.68/21.67 ppl versus 21.57 at rank 8; they do not test a larger weight-only channel. At matched tokens, the single-channel point estimates favor input space (21.44 vs. 23.27 validation ppl at 25B), with auxiliary parameters differing by $\sim 65\times$ (Appendix E, Takeaway 1). The width dose-response is non-monotone at 135M (Appendix E, takeaway 2). At 906M, the matched both-channel comparison improves from 13.80 to 13.71 ppl when S-PLE width rises from 64 to 128 (Section 4.4); this single-seed point estimate does not establish saturation.

J.1 Weight-only rank sweep and single-channel references

The 135M weight-only rank sweep trains PLT with S-PLE disabled and zFLoRA rank $r \in \{8, 16, 32\}$. These three runs share the training recipe, seed, corpus, and validation set; rank is the intended intervention. We compare their observed validation NLL on the common 0.5B-token grid through a fixed 50B snapshot. The channel-free PLT baseline and S-PLE-only widths 32/64/128 use the same trunk and validation set and are drawn from separate training runs; the controlled rank sweep comprises the three weight-only runs. The plotted CSV and figure are generated from per-run `metrics.jsonl` by `scripts/plot_weight_only_rank_sweep.py`.

At 50B, the baseline and weight-only ranks 8/16/32 have NLL 3.0980/3.0893/3.0844/3.0804, respectively. The S-PLE-only widths 32/64/128 are at 3.0089/3.0022/3.0506 (Figure 2). Thus rank 32 improves on rank 8 by 0.0089 nll, a modest nonzero gain, yet remains 0.0715 nll above S-PLE32. All three input-only widths outperform all three weight-only ranks at this snapshot, and width 128 is worse than widths 32/64 despite its larger table budget. The practical comparison therefore extends beyond one small adapter versus one large table configuration (Section 4.4). This does not isolate placement from size: rank 32 has approximately 11.8M adapter parameters versus S-PLE32's 95.5M. The cross-family gap is descriptive, not a capacity-matched effect; the runs are single-seed and the sweep is partial. The evidence supports the observed channel ordering, not a final-100B rank ordering or a general rank-saturation law.

J.2 Quality–latency view

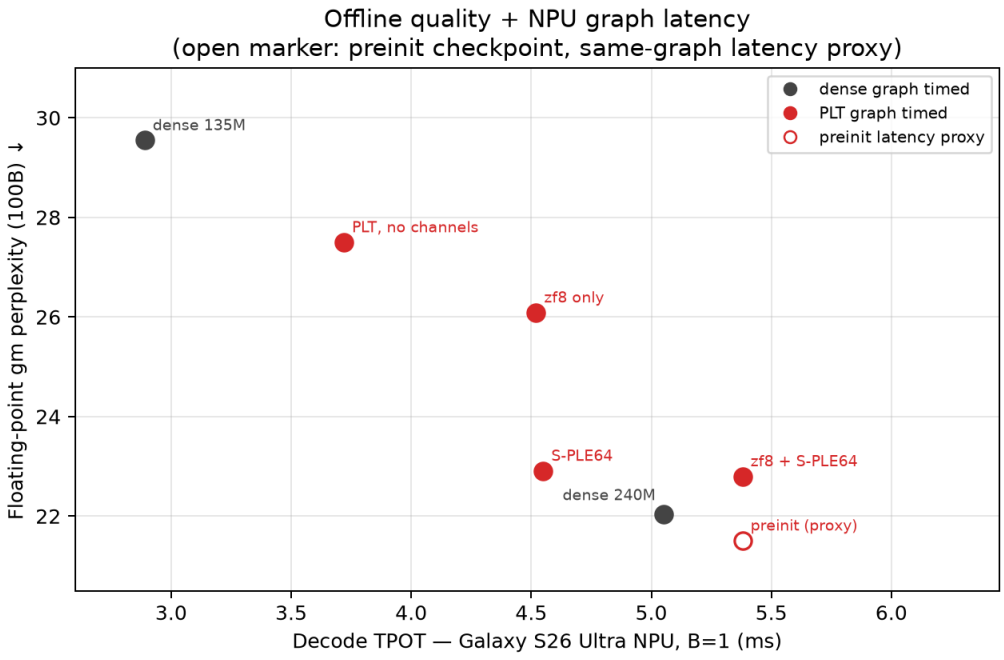


Figure 7: Quality and latency measured separately: 135M floating-point gm ppl at 100B paired with architecture-matched Galaxy S26 Ultra NPU decode TPOT (B=1, Table 22). The open preinit marker reuses the both-channel graph's timing; none of these points reports on-device quality. Total storage differs across arms and is reported in Table 4; this is not a fixed-storage frontier.

The figure includes the channel-free and weight-only NPU measurements. S-PLE64 has substantially lower floating-point perplexity than weight-only zFLoRA8 at similar measured graph decode latency, with much larger resident storage. The preinitialized checkpoint

improves quality on the both-channel topology but remains slower than dense-240M. No GPU-prefill or cloud-serving advantage is inferred from these NPU measurements.

K NPU ARM-LEVEL LATENCY AND NOMINAL BYTE BUDGET

Table 22: Measured on-device latency (Galaxy S26 Ultra, Hexagon HTP; all-integer w4a16-kv8 graphs; 1536-token prefill, 512 generated tokens, B=1; mean over repeated invocations). Prefill tok/s is the rate over the whole prompt; TPOT = per-token decode time; e2e estimates graph execution by summing mean prefill and decode calls, including the looped pipeline's trailing prefill call. Binary = compiled generation context, excluding host-side embedding tables; it is not total storage. The `zf8+sp64` graph has the same topology for scratch, pre-initialized, and cooldown checkpoints; its timing is an architecture-level proxy for the latter two, not a separate measurement.

Model	prefill tok/s	TPOT (ms)↓	e2e (s)↓	binary (MB)
SmolLM2-135M (dense, 30L)	34.5k	2.89	1.524	73.9
PLT-L2-135M, no channels	11.2k	3.72	2.042	73.0
LooPiT-L2-135M + `sp64`	10.1k	4.55	2.482	76.8
PLT-L2-135M + `zf8` only	6.5k	4.52	2.550	80.5
base-240M (dense, 2× depth, 60L)	18.1k	5.05	2.670	132.5
LooPiT-L2-135M + `zf8+sp64`	6.2k	5.38	3.002	87.2
base-840m (dense, 12L)	14.9k	10.08	5.264	487.6
PLT-840M, no channels	4.6k	10.89	5.909	463.7
LooPiT-840M + `sp64` (A1)	4.5k	11.37	6.162	470.7
PLT-840M + `zf8` only	2.2k	11.87	6.761	477.2
LooPiT-840M + `zf8+sp64` (T3)	2.2k	11.86	6.762	490.5
SmolLM2-1.7B (dense, 24L)	7.0k	19.12	10.008	923.9

Table 23: Nominal byte budget at w4a16-kv8 and 2048-token context, from the parameter censuses of Tables 6, 7 and 13. Weights: 0.5 B/param (int4; group scales omitted, <2%). Tables: int4 / fp16. KV: $2 \times n_{\text{layers}} \times n_{\text{kv}} \times d_{\text{head}} \times 2048$ B at int8, plus GSWA rings $2 \times n_{\text{layers}} \times n_{\text{kv}} \times d_{\text{head}} \times 128 \times (L-1)$ B. 135m shapes: 30L, 3 KV heads, d_{head}=64; 840m: 12L, 32 KV heads, d_{head}=64. Total uses int4 tables.

Arm	Weight bytes (int4)	Tables (int4 / fp16)	KV + rings	Total
SmolLM2-135M (30L)	67 MB	—	24 MB	91 MB
PLT-L2-135M, no channels (estimate)	67 MB	—	25 MB	92 MB
LooPiT-L2-135M + `sp64`	68 MB	94 / 377 MB	25 MB	187 MB
base-240M (60L)	121 MB	—	47 MB	168 MB
LooPiT-L2-135M + `zf8+sp64`	70 MB	94 / 377 MB	25 MB	189 MB
base-840m (12L)	453 MB	—	101 MB	554 MB
LooPiT-840M + `sp64` (A1)	455 MB	38 / 151 MB	107 MB	600 MB
LooPiT-840M + `zf8+sp64` (T3)	458 MB	38 / 151 MB	107 MB	603 MB
SmolLM2-1.7B (24L)	856 MB	—	201 MB	1057 MB

Table 22 reports on-device latency measured on a Samsung Galaxy S26 Ultra executing on the Qualcomm SM8850 Hexagon Tensor Processor (HTP), from all-integer w4a16-kv8 graphs compiled to QNN context binaries with QAIRT 2.49.40.260810: 4-bit per-output-channel weights, 16-bit asymmetric activations, an 8-bit symmetric KV cache, and 16×8 matrix multiplies, exported at fully static shapes with a context length of 2048. The converter reports no floating-point arithmetic fallback, with mask-comparison nodes counted separately; their backend fusion is inferred in the benchmark report, not independently established here. Calibration uses dummy zero-valued embedding inputs and real causal masks. Consequently, these are latency graphs, not accuracy-calibrated models. Host-side embedding/RoPE preparation and sampling are outside the reported graph-call timings; compiled context sizes likewise exclude host-resident tables. The looped rows run the deployment export path; dense rows are architecture-matched dense references. The `zf8+sp64` row times a graph topology also

used by the pre-initialized and cooldown checkpoints; it is an architecture-level latency proxy for those checkpoints, not a separate measurement.

Workload and graph schedules. Every row measures a 1536-token prompt followed by 512 generated tokens at batch 1. The dense baselines prefill 256 tokens per invocation; the looped model uses a dual-slot prefill graph that evaluates loop 1 on the current chunk and loop 2 on the preceding chunk concurrently, advancing 128 new tokens per invocation and requiring one trailing invocation to drain the pipeline, which the e2e column counts (6 invocations for dense, 13 for looped). Both decode one token per invocation. The prefill tok/s column divides the 1536-token prompt by the end-to-end prefill time — not the per-call rate the harness reports for the dual graphs.

Thermal and clock control; reproducibility. Measurements are taken with the thermal engine disabled, both CPU clusters pinned to the performance governor at maximum frequency, and the NPU clock voted at burst, with thermal zones read before and after each run to confirm no throttling. We report the mean over repeated invocations; rebuilding an unchanged graph reproduces to within 0.17 ms.

What the measured rows identify. The 2×2 ablation prices each channel against the channel-free loop graph under one recipe. At 135m, `sp64` adds +0.83 ms/token (4.55 vs. 3.72) and `zf8` adds +0.80 ms (4.52); both together add +1.66 ms (5.38), the sum of parts (+1.63) to within noise. At 906M the decode increments are +0.48 ms (`sp64`, 11.37 vs. 10.89) and +0.98 ms (`zf8`, 11.87), sub-additive — the both-channel graph adds +0.97 ms (11.86), below the +1.46 ms sum of parts. Arm-level premiums over same-size dense are +57/+86% TPOT at 135m and +13/+18% at 906M; latency alone does not measure on-chip residency or DRAM traffic. On the prefill side, the looped graphs trade rate for the dual-slot schedule (11.2k vs. 34.5k tok/s at 135m against same-size dense), and `zf8` roughly halves the 840m prefill rate (4.6k to 2.2k tok/s); prefill-side optimization remains open, while the loop arms stay 38–43% faster in decode than the dense 1.7B reference. The channel-free 135m graph's trained counterpart is the no-channel baseline of Table 1, so its latency advantage over base-240M is an architecture-level latency result, not a quality claim.

Nominal byte budget. Table 23 estimates packed static weights, token tables, and KV storage from parameter counts and tensor shapes; it is not a measurement of per-token DRAM traffic or NPU memory residency. The loop arms reuse one backbone weight set; S-PLE tables add stored bytes but gather one row per (layer, loop) and token. At 135m, `sp64` has 162 MB of static weights including int4 tables (445 MB with fp16 tables), versus 121 MB for the dense 2×-depth reference. At 906M, A1 has 493/606 MB under the same table policies, versus 856 MB for the dense 1.7B reference. Including the estimated KV cache, the int4 totals are 187 vs. 168 MB at 135m and 600 vs. 1057 MB at 906M. Thus the quality–latency result does not imply a smaller footprint at 135m.

The earlier analytic cost model is not used for deployment claims. It reproduced the dense depth-scaling ratio (1.79× modeled vs. 1.75× measured for 240m/135m TPOT) but underestimated absolute TPOT by approximately 2×. Its +1.4% estimate for adding S-PLE to a loop graph undershoots the measured channel increment (+0.83 ms/token, +22% over the channel-free loop at 135m). Deployment claims rest on Table 22 only.