

Greedy Local Learning for Language Model Pretraining: Gaps and Objective Design

Jihwan Moon*

INFOCZ Inc.

Seoul National University

jihwan.moon@vision.snu.ac.kr

Sheir A. Zaheer

KC ML2

sheir@kc-ml2.com

Jinmyoung Lee

KC ML2

mys1458@kc-ml2.com

Gunhee Kim

Seoul National University

gunhee@snu.ac.kr

Chan Y. Park

KC ML2

chan.y.park@kc-ml2.com

Abstract

Greedy block-wise local learning splits a network into gradient-isolated blocks trained by local auxiliary losses, deleting the backward pass between blocks: inter-stage communication becomes forward-only and every block can step its optimizer independently, properties directly relevant to decentralized model-parallel training. Local learning is competitive with end-to-end backpropagation on image classification, and on small Transformers it is known to trade a worse best loss for parallel speedup. How this loss gap behaves in autoregressive language model (LM) pretraining at larger scale, and which auxiliary designs reduce it, has not been measured. We present a token-budget-matched empirical study at 125M and 400M parameters with $K \in \{1, 2, 4\}$ blocks at Chinchilla-optimal budgets, factorizing the auxiliary design into network architecture and training objective. We observe: (i) the gap to end-to-end training more than doubles from $K=2$ to $K=4$, but at $K=4$ shrinks from 125M to 400M; (ii) replacing an MLP auxiliary with a Transformer-based one is a strong network-side intervention, recovering 22–41% of the gap; (iii) a multi-token-prediction (MTP) auxiliary objective helps at the first block boundary, whereas adding it at deeper boundaries hurts, and restricting it to the first block yields the best $K=4$ configuration (+0.062 vs. +0.075 nats at 400M); and (iv) deployment-style per-block execution reduces activation memory by up to $2.2\times$. We frame these results as an empirically grounded method direction rather than a finalized method: local objectives should apply future-predictive pressure selectively across boundaries while resisting shortcuts that bypass predictive content.

1 Introduction

Interest in training large language models across many loosely connected, memory-constrained workers has grown rapidly, from low-communication data-parallel schemes [Douillard et al., 2023, Jaghouar et al., 2024a] to collaborative pretraining runs spanning the globe [Jaghouar et al., 2024b]. The complementary axis, *model parallelism* over weak workers where no single device can hold the model, remains harder: pipeline-parallel training must ship both forward activations and backward gradients across stage boundaries, while the backward pass couples the stages’ schedules and updates [Ryabinin et al., 2023, Ramasinghe et al., 2025].

*Work done while at KC ML2.

Greedy block-wise local learning [Belilovsky et al., 2019, 2020] is a natural candidate for this setting. The network is split into K gradient-isolated blocks; each block is trained by an auxiliary head with a local loss, and no error signal crosses block boundaries. Forward activations still flow between stages, but activation gradients do not. In a stage-partitioned deployment, this makes inter-stage traffic forward-only, removes backward update dependencies, and limits each worker’s backward graph to one block and its auxiliary head. A line of work has made local learning competitive with end-to-end backpropagation on image classification [Belilovsky et al., 2019, Pyeon et al., 2021, Wang et al., 2021, Su et al., 2024, Ma et al., 2024]. For language, Laskin et al. [2021] trained small Transformers (6M parameters in their main experiments) with local updates and found that they reach a target loss with less sequential compute, but do not reach as low a loss as backpropagation within a fixed example budget. How that gap behaves in autoregressive language model (LM) pretraining at 125M–400M parameters under the standard next-token-prediction (NTP) objective, judged by held-out likelihood, and which auxiliary designs reduce it, has, to our knowledge, not been measured.

Two established accounts motivate our measurements. In restricted function classes, the *backward feature correction* account shows that joint training with upper layers corrects lower-layer feature errors, a signal that gradient isolation removes [Allen-Zhu and Li, 2020]. The representation-side account argues that a greedy local loss can discard information needed downstream [Wang et al., 2021]. We ask how these effects appear in LM pretraining: how the likelihood gap scales with block count and model size, and which aspects of auxiliary design reduce it.

We split 125M- and 400M-parameter Llama-style Transformers [Vaswani et al., 2017, Touvron et al., 2023a] into $K \in \{1, 2, 4\}$ equal-depth blocks and train at Chinchilla token budgets [Hoffmann et al., 2022]. We factor auxiliary design into the network placed after a block and the objective used to train it. On the network side, we compare an MLP head with a Transformer head. On the objective side, we compare NTP with MTP, whose targets are already present in the token stream and add no inter-stage communication. Per-block losses diagnose whether successive blocks improve the representations they inherit, and deployment-style measurements quantify the memory consequence of isolating their backward graphs.

Our contributions:

- A token-budget-matched likelihood study of greedy local learning for LM pretraining across model scale and block count, taking the small-Transformer language-modeling results of Laskin et al. [2021] to 125M–400M parameters and Chinchilla-optimal budgets.
- Per-block evidence consistent with *representation myopia*: later blocks barely improve their inherited representation under local NTP supervision.
- A factorized comparison showing that a Transformer-based auxiliary is a strong network-side intervention, while MTP is beneficial at the first boundary but not when added at deeper boundaries. Together with the NTP+CPC failure [van den Oord et al., 2018, Löwe et al., 2019], these experiments suggest a design direction for boundary-adaptive, shortcut-resistant local objectives.
- Deployment-style memory measurements and analytical accounting of communication and auxiliary training compute for local learning as decentralized model parallelism. We release the factorized implementations used for these comparisons.

2 Related work

Greedy block-wise methods were developed primarily for image classification. DGL established competitive locally trained CNNs [Belilovsky et al., 2019, 2020]; SEDONA searches decoupling points and auxiliary architectures [Pyeon et al., 2021]; and InfoPro adds reconstruction pressure to preserve downstream-useful information [Wang et al., 2021]. MAN and AugLocal incorporate copies of later computation into local paths [Su et al., 2024, Ma et al., 2024], which introduces synchronization or replicated downstream computation in a decentralized pipeline. Closest to our setting, Laskin et al. [2021] study *local parallelism* on MLPs, ResNets, and small Transformer language models. They vary the partition under one auxiliary form and report compute–walltime trade-offs, whereas we measure the likelihood gap at Chinchilla token budgets across scale and auxiliary design. Two recent LM studies address different settings: LoPT locally post-trains an already pretrained model, with lower blocks reconstructing detached embeddings rather than receiving an

LM pretraining signal [Shi et al., 2026]; DiffusionBlocks replaces likelihood training with per-block denoising [Shing et al., 2026].

Decentralized LM training spans two complementary axes. Local-update data-parallel methods reduce synchronization between model replicas [Douillard et al., 2023, Jaghouar et al., 2024a, Douillard et al., 2024, Jaghouar et al., 2024b, Peng et al., 2024]. Model-parallel approaches instead distribute one model across unreliable or weakly connected workers [Ryabinin et al., 2023, Borzunov et al., 2023] or compress boundary activations and gradients [Ramasinghe et al., 2025, Janson et al., 2026]. Our study lies on the second axis: local losses delete the backward boundary traffic and update dependency, and can compose with the first axis. An extended review appears in Appendix A.

3 Study design

3.1 Greedy block-wise local learning

Consider a decoder-only Transformer with token embedding E , L Transformer layers, and an output head (final norm and vocabulary projection). We partition the L layers into K contiguous, equal-depth blocks $F_1, \dots, F_K$. Writing $\text{sg}[\cdot]$ for the stop-gradient operator, the forward pass over a token sequence $x_{1:T}$ is

$$h_0 = E(x_{1:T}), \quad h_1 = F_1(h_0), \quad h_k = F_k(\text{sg}[h_{k-1}]), \quad k = 2, \dots, K, \quad (1)$$

so activation values are identical to the original network, but no gradient crosses a block boundary. Each non-final block $k < K$ is trained by an *auxiliary network* g_k that maps the block output h_k to a prediction, scored by a local *auxiliary loss* $\mathcal{L}_k = \ell_k(g_k(h_k), x)$; the final block keeps the model’s real output head and the standard NTP cross-entropy $\mathcal{L}_K$. Training minimizes $\sum_{k=1}^K \mathcal{L}_k$, and by Eq. (1) the parameters of (F_k, g_k) receive gradient from $\mathcal{L}_k$ alone (the embedding is trained by $\mathcal{L}_1$). For $K=1$ the formulation reduces exactly to standard end-to-end training.

Gradient isolation is what makes this formulation fit model parallelism. In a pipelined deployment, worker k holds (F_k, g_k) , receives h_{k-1} , and updates from $\mathcal{L}_k$ alone, so boundary traffic is forward-only and stages have no backward update dependency (Appendices D and E).

This leaves two design choices per block, the auxiliary network g_k and the auxiliary objective ℓ_k , which the next two subsections vary.

3.2 Auxiliary networks: MLP-aux and Transformer-aux

We first vary g_k while holding the local objective fixed to NTP. MLP-aux applies a pre-normalized SwiGLU FFN with a residual connection, followed by RMSNorm and a vocabulary projection. Transformer-aux prepends one Transformer block to the same head, adding local self-attention and depth. Both heads are untied across blocks, trained only through their local NTP losses and discarded at inference.

3.3 Auxiliary objective: multi-token prediction

We next vary ℓ_k using MTP [Gloeckle et al., 2024, DeepSeek-AI, 2024]. An MTP-equipped auxiliary head first applies the same Transformer block as Transformer-aux. Its output u_k then enters an offset-specific residual FFN,

$$z_{k,j} = u_k + \text{FFN}_{k,j}(\text{RMSNorm}_{k,j}(u_k)), \quad j \in \{1, 2\}, \quad (2)$$

where $j = 1$ predicts x_{t+1} and $j = 2$ predicts x_{t+2} . The branches share the final RMSNorm and vocabulary projection, following the shared-head design of DeepSeek-V3. The $j = 1$ branch is wiring-identical to Transformer-aux; removing the $j = 2$ branch recovers that control exactly. The added branch costs one FFN and norm in parameters, and in training compute it also requires a second pass through the shared vocabulary projection.

We compare *first-block* MTP (Transformer-aux + MTP), which applies equally weighted two-offset prediction only at the first auxiliary block, with *all-block* MTP, which applies a depth-weighted two-offset head at every auxiliary block (Appendix B). They coincide at $K=2$. The final block remains NTP-only, and shifted token targets add no inter-stage communication.

Table 1: Validation NLL, absolute for end-to-end ($K=1$) and as Δ against it for $K \in \{2, 4\}$ (nats/token; lower is better). Transformer-aux + MTP denotes first-block MTP; the all-block comparison is in Table 2.

	MLP-aux	Transformer-aux	Transformer-aux + MTP
<i>125M-class</i>	(end-to-end $K=1$: 2.771)		
$K=2$ (Δ)	+0.046	+0.028	+0.017
$K=4$ (Δ)	+0.132	+0.078	+0.074
<i>400M-class</i>	(end-to-end $K=1$: 2.418)		
$K=2$ (Δ)	+0.041	+0.032	+0.027
$K=4$ (Δ)	+0.112	+0.075	+0.062

3.4 Experimental setting

Models and data. We train Llama-architecture [Touvron et al., 2023a] decoder-only Transformers at two sizes: a 125M-class model ($d_{\text{model}}=768$, 12 layers, 12 heads; 85M non-embedding parameters) and a 400M-class model ($d_{\text{model}}=1024$, 24 layers, 16 heads; 308M non-embedding parameters), with the Llama-2 32k SentencePiece tokenizer [Touvron et al., 2023b]. We use the FineWeb-Edu 10BT sample [Penedo et al., 2024], packed into sequences of 4,096 tokens.

Training. All runs use a global batch of 2^{19} ($\approx 524k$) tokens per step and strict Chinchilla-optimal budgets of 20 tokens per total parameter (embeddings included, counted at $K=1$): 5,100 steps ($\approx 2.7B$ tokens) for the 125M class and 14,300 steps ($\approx 7.5B$ tokens) for the 400M class. Hyperparameters are shared across methods and block counts within each scale, and all quality experiments are synchronous simulations with the same seed. Each method is trained with $K \in \{1, 2, 4\}$ equal-depth blocks. The $K=1$ configuration is bit-identical to standard end-to-end training and serves as the baseline. Optimizer, schedule, precision, and checkpoint details are in Appendix B.

Evaluation. The headline metric is held-out validation NLL per token on FineWeb-Edu, always computed through the final block’s real NTP head, so it is directly comparable across methods, block counts, and to end-to-end training.

4 Observations

4.1 Observation 1: the isolation gap grows with block count

Table 1 reports the headline comparison: validation NLL against the shared end-to-end baseline for the three auxiliary designs at both scales. Figure 2 shows the corresponding 400M validation curves.

For every auxiliary design, the gap more than doubles when the block count increases from $K=2$ to $K=4$ (e.g. MLP-aux at 125M: $+0.046 \rightarrow +0.132$), the direction Laskin et al. [2021] report for small Transformers. At $K=4$, moving from 125M to 400M reduces the gap for all three designs (MLP-aux $+0.132 \rightarrow +0.112$, Transformer-aux $+0.078 \rightarrow +0.075$, and first-block MTP $+0.074 \rightarrow +0.062$). This scale trend is specific to the deeper partition: at $K=2$, two of the three gaps increase slightly with scale. At the lowest-gap operating point (first-block MTP, $K=2$, 400M) the cost of gradient isolation is $+0.027$ nats, about 1.1% relative validation NLL.

4.2 Observation 2: local losses exhibit representation myopia

Figure 1 tracks each block’s local loss at 125M, $K=4$. Under both NTP auxiliaries the four curves collapse onto nearly the same trajectory: the first block already reaches almost the loss of the full stack, and blocks above it barely improve on the representation they inherit. Each block behaves as a standalone next-token predictor rather than a feature extractor for its successors. We call this pattern *representation myopia*. The collapse is evidence consistent with locally useful features failing to improve downstream, rather than direct proof that particular information has been discarded. The same qualitative pattern holds at 400M (Appendix Figure 3).

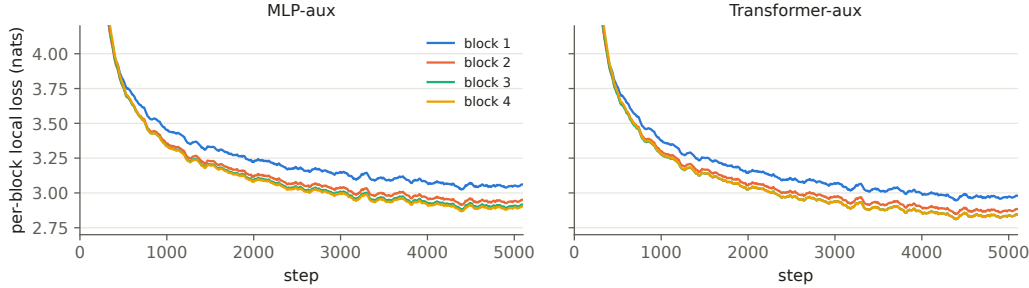


Figure 1: Per-block local losses at 125M, $K=4$, under MLP-aux (left) and Transformer-aux (right). The curves collapse: blocks after the first barely improve on the representation they inherit, and a larger head lowers all curves without separating them.

Table 2: MTP placement at $K=4$: Δ validation NLL against end-to-end (nats/token).

Configuration	125M	400M
Transformer-aux (NTP only)	+0.078	+0.075
first-block MTP	+0.074	+0.062
all-block MTP, linear	+0.082	+0.086

Table 3: Per-offset training CE for all-block MTP (125M, $K=4$, last 100 logged steps).

Block	w_{t+2}	CE($t+1$)	CE($t+2$)
1	0.75	3.087	4.588
2	0.50	2.929	4.500
3	0.25	2.858	4.518

4.3 Observation 3: the Transformer-based auxiliary helps most

Replacing MLP-aux with Transformer-aux reduces the gap by 39% and 41% at 125M ($K=2, 4$), and by 22% and 33% at 400M. The per-block curves also move downward without separating, suggesting that the stronger auxiliary makes locally predictive features easier to decode but does not remove the myopic training pattern. Because the two heads are not parameter- or compute-matched, we interpret this as an architecture-level comparison, not as a controlled causal estimate of attention alone.

4.4 Observation 4: MTP depends on boundary position

Table 2 makes the placement comparison explicit. At $K=2$, the single auxiliary boundary is necessarily the first one, and MTP improves over Transformer-aux by 0.011 nats at 125M and 0.005 at 400M. At $K=4$, first-block MTP again improves over Transformer-aux, whereas equipping all auxiliary blocks with MTP is worse than Transformer-aux at both scales. Thus, the directly supported statement is that MTP helps at the first boundary, while adding the same pressure at deeper boundaries hurts. The 400M first-block margin over Transformer-aux is 0.013 nats, about five times the measured ~ 0.0025 -nat run-to-run noise floor; the 125M margin (0.004) is only consistent in direction. Differences between linear and uniform all-block weighting are near that noise floor (Appendix G).

The per-offset diagnostic in Table 3 shows the corresponding trade-off inside all-block MTP. As the second-token weight decreases across successive auxiliary blocks, next-token CE improves from 3.087 to 2.858. The lower blocks therefore give up more immediate predictive sharpness under the stronger forward-looking pressure. This mechanism is consistent with first-boundary MTP helping while the same pressure becomes counterproductive closer to the final head. We hypothesize that early MTP preserves broad future-predictive information that the many remaining layers can exploit. Near the final head, the more uncertain second-token target instead competes with NTP, with too little downstream depth to repay the loss in immediate precision.

Beyond MTP, we also tested an NTP+CPC hybrid auxiliary at 125M, $K=2$. It increased the gap to +0.154 nats, versus +0.028 for Transformer-aux and +0.017 for MTP, because position and document identity satisfied InfoNCE without future-predictive content (Appendix H).

4.5 Observation 5: local execution reduces memory at a training-compute cost

Table 4 reports peak GPU memory per worker, measured on a single GPU at batch $4 \times 4,096$ tokens with bf16 autocast: the footprint of one pipeline stage, i.e., gradients, optimizer state, and activations for one block plus its auxiliary head. For the MTP head we adopt the sequential per-offset

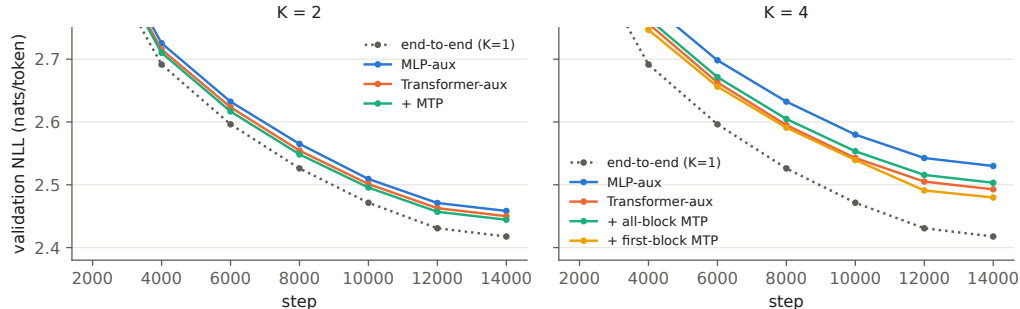


Figure 2: Validation NLL at 400M for $K=2$ (left) and $K=4$ (right). At $K=4$ the first-block placement improves over both Transformer-aux and the all-block MTP weighting throughout training.

computation of Gloeckle et al. [2024] as the default execution: each offset branch runs its forward and backward eagerly, so only one vocabulary-sized logits tensor is alive at a time. Splitting the 400M model into $K=4$ blocks cuts peak memory from 29.2 to 14.5–15.8 GiB (~ 1.9 – $2.0\times$) and activation memory from 27.7 to 12.4–13.4 GiB (up to $2.2\times$), depending on the auxiliary head. The reduction is sublinear in K because every worker still runs a vocabulary-sized auxiliary head; the Transformer-block activations themselves scale as $1/K$. Transformer-aux + MTP carries the MTP head only at the first stage and plain Transformer-aux heads elsewhere, and under sequential execution its maximum-stage footprint at $K=4$ is within 0.1 GiB of Transformer-aux. Per-stage values are in Appendix B.

Table 4: Peak/activation GPU memory per worker (GiB; single GPU, batch $4 \times 4,096$, bf16). Non-E2E entries hold one block and its auxiliary head and report the largest stage. MTP uses sequential per-offset computation [Gloeckle et al., 2024].

Method	125M-class		400M-class	
	$K=2$	$K=4$	$K=2$	$K=4$
End-to-end ($K=1$)	14.50/13.98		29.16/27.68	
MLP-aux	11.03/10.40	9.52/8.56	19.07/17.43	14.52/12.41
Transformer-aux	12.05/11.07	10.32/9.25	20.49/18.33	15.67/13.36
Transformer-aux + MTP (first block only)	12.37/11.24	10.36/9.26	21.02/18.56	15.76/13.37

Compute overhead. Auxiliary heads are discarded at inference but add training compute, which our token-matched comparisons do not charge (Appendix F). One head costs 19%, 26%, and 45% of end-to-end training FLOPs at 125M for MLP-aux, Transformer-aux, and an MTP head, and 9%, 13%, and 22% at 400M. Most of this is the vocabulary projection that every local head needs, which an MTP head runs twice. At $K=4$ the totals are 78% for Transformer-aux and 97% for first-block MTP at 125M, and 39% and 48% at 400M.

5 Conclusion

We measured greedy block-wise local learning for LM pretraining at Chinchilla budgets and found five things. The gap to end-to-end training more than doubles from two to four blocks but shrinks with scale at $K=4$. Collapsed per-block losses are consistent with representation myopia. The Transformer-based auxiliary is an effective network-side intervention. MTP helps at the first boundary but hurts at deeper ones, so first-block MTP is the best $K=4$ configuration at both scales. Per-block execution reduces activation memory by up to $2.2\times$ at an added training-compute cost.

Limitations: our scale is capped at 400M parameters, each configuration uses a single seed, and comparisons are token-matched rather than compute-matched. We offer measurements, not yet a concrete method. The contrast between boundary-selective MTP and shortcut-prone NTP+CPC suggests that future local objectives should preserve future-predictive content, resist position and document-identity shortcuts, and adapt their pressure to the position of each block boundary.

References

- Z. Allen-Zhu and Y. Li. Backward feature correction: How deep learning performs deep (hierarchical) learning. *arXiv preprint arXiv:2001.04413*, 2020.
- Anonymous. LIFT: Efficient layer-wise fine-tuning for large model models. OpenReview preprint u0INlprg3U, 2023.
- E. Belilovsky, M. Eickenberg, and E. Oyallon. Greedy layerwise learning can scale to ImageNet. In *International Conference on Machine Learning*, 2019.
- E. Belilovsky, M. Eickenberg, and E. Oyallon. Decoupled greedy learning of CNNs. In *International Conference on Machine Learning*, 2020.
- A. Borzunov, D. Baranchuk, T. Dettmers, M. Ryabinin, Y. Belkada, A. Chumachenko, P. Samygin, and C. Raffel. Petals: Collaborative inference and fine-tuning of large models. In *ACL System Demonstrations*, 2023.
- DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.
- A. Douillard, Q. Feng, A. A. Rusu, R. Chhaparia, Y. Donchev, A. Kuncoro, M. Ranzato, A. Szlam, and J. Shen. DiLoCo: Distributed low-communication training of language models. *arXiv preprint arXiv:2311.08105*, 2023.
- A. Douillard, Q. Feng, A. A. Rusu, A. Kuncoro, Y. Donchev, R. Chhaparia, I. Gog, M. Ranzato, J. Shen, and A. Szlam. DiPaCo: Distributed path composition. *arXiv preprint arXiv:2403.10616*, 2024.
- F. Gloeckle, B. Y. Idrissi, B. Rozière, D. Lopez-Paz, and G. Synnaeve. Better & faster large language models via multi-token prediction. In *International Conference on Machine Learning*, 2024.
- J. Hoffmann et al. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, 2022.
- M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, and K. Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *International Conference on Machine Learning*, 2017.
- S. Jaghouar, J. M. Ong, and J. Hagemann. OpenDiLoCo: An open-source framework for globally distributed low-communication training. *arXiv preprint arXiv:2407.07852*, 2024a.
- S. Jaghouar et al. INTELLECT-1 technical report. *arXiv preprint arXiv:2412.01152*, 2024b.
- P. Janson, E. Oyallon, and E. Belilovsky. Learned subspace compression for communication-efficient pipeline parallelism. In *ICML Workshop on Connecting Low-Rank Representations in AI*, 2026.
- C. H. Koneputugodage, T. Ajanthan, S. Ramasinghe, H. M. Dolatabadi, S. Siriwardhana, G. Avraham, V. Shevchenko, K. Pajak, J. Snegin, and A. Long. Factored gossip DiLoCo: Reducing blocking communication in DiLoCo. In *International Conference on Machine Learning*, 2026.
- M. Laskin, L. Metz, S. Nabarro, M. Saroufim, B. Noune, C. Luschi, J. Sohl-Dickstein, and P. Abbeel. Parallel training of deep networks with local updates. *arXiv preprint arXiv:2012.03837*, 2021.
- S. Löwe, P. O’Connor, and B. S. Veeling. Putting an end to end-to-end: Gradient-isolated learning of representations. In *Advances in Neural Information Processing Systems*, volume 32, 2019.
- C. Ma, J. Wu, C. Si, and K. C. Tan. Scaling supervised local learning with augmented auxiliary networks. In *International Conference on Learning Representations*, 2024.
- R. Pan, X. Liu, S. Diao, R. Pi, J. Zhang, C. Han, and T. Zhang. LISA: Layerwise importance sampling for memory-efficient large language model fine-tuning. In *Advances in Neural Information Processing Systems*, 2024.
- G. Penedo et al. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.
- B. Peng, J. Quesnelle, and D. P. Kingma. DeMo: Decoupled momentum optimization. *arXiv preprint arXiv:2411.19870*, 2024.
- M. Pyeon, J. Moon, T. Hahn, and G. Kim. SEDONA: Search for decoupled neural networks toward greedy block-wise learning. In *International Conference on Learning Representations*, 2021.

- S. Ramasinghe, T. Ajanthan, G. Avraham, Y. Zuo, and A. Long. Subspace networks: Scaling decentralized training with communication-efficient model parallelism. In *Advances in Neural Information Processing Systems*, 2025.
- M. Ryabinin, T. Dettmers, M. Diskin, and A. Borzunov. SWARM parallelism: Training large models can be surprisingly communication-efficient. In *International Conference on Machine Learning*, 2023.
- H. Shi, T. Han, P. Wang, Z. Wang, X. Yang, and J. Su. Rethinking local learning: A cheaper and faster recipe for LLM post-training. *arXiv preprint arXiv:2605.04913*, 2026.
- M. Shing, M. Koyama, and T. Akiba. DiffusionBlocks: Block-wise neural network training via diffusion interpretation. In *International Conference on Learning Representations*, 2026.
- J. Su et al. Momentum auxiliary network for supervised local learning. In *European Conference on Computer Vision*, 2024.
- H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.
- H. Touvron, L. Martin, K. Stone, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.
- A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.
- Y. Wang, Z. Ni, S. Song, L. Yang, and G. Huang. Revisiting locally supervised learning: An alternative to end-to-end training. In *International Conference on Learning Representations*, 2021.

A Extended related work

A.1 Greedy block-wise local learning

Greedy block-wise local learning has mainly been studied in the context of image classification. Decoupled greedy learning (DGL) [Belilovsky et al., 2019, 2020] established that CNNs split into gradient-isolated, locally supervised blocks can approach end-to-end accuracy on ImageNet; SEDONA [Pyeon et al., 2021] searches over decoupling points and auxiliary architectures, finding the split decision more critical than the auxiliary choice and “finding appropriate lower blocks” key to the success of greedy block-wise learning; InfoPro [Wang et al., 2021] augments each block’s local task loss with a reconstruction term so that forwarded features retain input information useful to later blocks, MAN [Su et al., 2024] augments each block’s local path with a momentum (EMA) copy of the next block’s first layer, and AugLocal [Ma et al., 2024] builds each layer’s auxiliary network from a uniformly sampled, depth-reduced subset of its subsequent layers.

Two lines of work explain the performance gap between end-to-end backpropagation and local learning. Allen-Zhu and Li [2020] prove, for restricted function classes, that layerwise training cannot match end-to-end learning because lower-layer feature errors are corrected only via joint training with higher layers (*backward feature correction*). InfoPro offers a complementary representation-side account: a greedy task loss can discard information needed downstream, providing evidence using decoder-based mutual-information estimates on CNN classifiers.

Laskin et al. [2021] is the most direct precedent for local updates in language modeling. They study *local parallelism* in three forms: greedy updates with one layer per block, *chunked* updates with several consecutive layers per block (our K -block partition), and *overlapping* updates, in which adjacent blocks share a boundary layer and average its two updates. Each block is trained by an auxiliary objective of the same form as the global one. For their Transformer, the auxiliary classifier is a layer norm and a vocabulary projection. They trace compute–walltime Pareto frontiers for MLPs on CIFAR-10, ResNet-50 on ImageNet, and Transformers on LM1B, all trained for a fixed number of examples. On a 4-layer, 6M-parameter Transformer, local methods reach a loose loss target with about $4\times$ less sequential compute and tighter targets with about $2\times$ less, the latter only with overlapping updates. They do not reach the same minimum within the example budget: the

best validation losses are 3.62 for backpropagation, 3.66 for overlapping, 3.73 for two blocks, and 3.76 for greedy updates. A larger 6-layer, 512-wide Transformer in their appendix gives 3.62 for backpropagation, 3.65 for two blocks, 3.68 for three blocks, 3.68 for overlapping, and 3.80 for greedy updates. They also show on ResNets that larger blocks generalize better, that locally trained networks fit random labels worse than backpropagation does, and that chunked local updates raise ImageNet throughput over pipelined backpropagation by 8–41% on real hardware despite the added auxiliary computation.

Our study is complementary. We train at 125M–400M parameters under Chinchilla token budgets and vary the auxiliary network and objective, which their study holds fixed. We therefore measure the likelihood gap as a function of K , scale, and auxiliary design, whereas they measure the compute–walltime frontier. For non-overlapping partitions, their loss gap grows from two to three blocks in both Transformers, in agreement with our Observation 1. They also raise, as a potential downside, that fully greedy updates “force the layers to learn features that are only relevant to their local objective”. The collapsed per-block losses of Observation 2 are consistent with it.

Two recent works bring block-wise training to language models, in settings different from ours. LoPT [Shi et al., 2026] applies local learning to *post-training*: the last block is trained using SFT or RL objective, while lower blocks reconstruct detached input embeddings. This presupposes a pretrained interface; from random initialization, the reconstruction target is itself random and supplies no language-modeling signal, so the objective does not transfer to pretraining from scratch. DiffusionBlocks [Shing et al., 2026] makes blocks fully independent by reinterpreting the residual stack as reverse diffusion and *replacing* the loss with per-block L2 denoising; the converted model is no longer a likelihood-trained LM and is evaluated only via MAUVE and judge-model generative perplexity, so its results are not comparable to an NLL-matched end-to-end baseline.

A.2 Layer-wise fine-tuning of LLMs

A neighboring family reduces *fine-tuning* cost by training only a subset of layers at a time: LIFT [Anonymous, 2023] fine-tunes a single Transformer layer per iteration, and LISA [Pan et al., 2024] importance-samples which layers to unfreeze. Both select which parameters to *update* under the single end-to-end loss at the real head; the forward and backward passes still traverse every layer, so they reduce fine-tuning memory rather than inter-stage communication or update locking, and they adapt pretrained models rather than pretrain from scratch.

A.3 Decentralized LLM pretraining

Decentralized pretraining has advanced along two composable axes. The *data-parallel, local-update* family reduces synchronization frequency or gradient traffic between model replicas: DiLoCo [Douillard et al., 2023] and its open replication and scaling [Jaghour et al., 2024a], path composition in DiPaCo [Douillard et al., 2024], gossip-based variants that reduce blocking communication [Koneputugodage et al., 2026], the INTELLECT-1 globally distributed 10B run [Jaghour et al., 2024b], and compressed optimizer states in DeMo [Peng et al., 2024].

The *model-parallel* family, where our work sits, splits the model across workers: SWARM parallelism [Ryabinin et al., 2023] and Petals [Borzunov et al., 2023] route through unreliable pipeline stages, while Subspace Networks [Ramasinghe et al., 2025] and learned subspace compression [Janson et al., 2026] compress inter-stage activation and gradient traffic. Notably, Ramasinghe et al. [2025] report that the backward direction is the harder one to compress due to error accumulation. Local learning is complementary: it deletes the backward direction entirely, leaving only forward traffic for compression and removing backward update dependencies. The two axes compose: a local-update scheme can run across replicas of a locally trained pipeline.

B Training and implementation details

All runs use AdamW with learning rate 3×10^{-3} , weight decay 0.033, gradient clipping at 1.0, and bf16 mixed precision. The learning rate follows a cosine decay to near zero, with 300 warmup steps at 125M and 1,000 at 400M. Final evaluations use the last saved checkpoint: step 5,000 for 125M and step 14,000 for 400M, so all reported comparisons are step-matched.

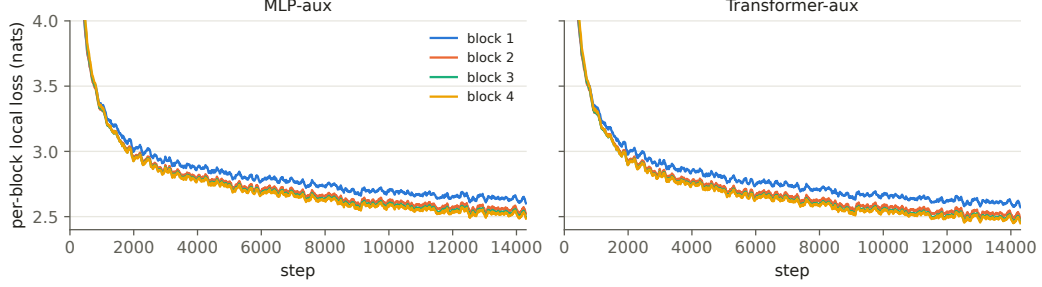


Figure 3: Per-block local losses at 400M, $K=4$, under MLP-aux (left) and Transformer-aux (right). As at 125M, later blocks provide only small improvements over the first block, and the Transformer-based auxiliary lowers the curves without separating them.

For all-block MTP, the k th auxiliary block (indexed from one) uses the linear schedule

$$\mathcal{L}_k = \frac{k}{K} \text{CE}_{k,1} + \left(1 - \frac{k}{K}\right) \text{CE}_{k,2}, \quad (3)$$

which places more second-token weight on earlier blocks. We also evaluate uniform weighting, $1/2$ per offset at every auxiliary block. At $K=2$ there is a single auxiliary block, so first-block MTP, all-block linear, and all-block uniform coincide at $1/2$ per offset.

The memory benchmark executes each stage separately. At 125M, $K=4$, the per-stage peaks are 10.14/10.21/10.32/9.36 GiB for Transformer-aux and 10.34/10.25/10.36/9.41 GiB for first-block MTP. At 400M they are 15.24/15.43/15.67/14.47 GiB and 15.55/15.51/15.76/14.56 GiB, respectively. MTP uses sequential per-offset execution, so only one vocabulary-sized logits tensor is live at a time.

C Per-block diagnostics at 400M

Figure 3 repeats the $K=4$ per-block diagnostic at 400M. The local losses again cluster tightly after the first block under both auxiliary networks, while the Transformer-based head shifts the curves downward without removing their collapse. The representation-myopia pattern therefore persists at the larger scale.

D Backward and update locking

Following Jaderberg et al. [2017], *forward locking* means that a module waits for its input from the preceding module; *update locking* means that it waits for the loss; and *backward locking* means that it waits for gradients to traverse the downstream modules. Pipeline backpropagation therefore gates each stage’s update on a forward–backward round trip.

Local losses remove update and backward locking: a stage can update after computing its own loss, without a signal from above. Forward locking remains, because stage k still consumes h_{k-1} . Stages can stream successive microbatches through this remaining forward dependency, although ordinary pipeline fill, drain, and load-imbalance effects remain.

E Analytical inter-stage communication

Table 5 gives the analytical inter-stage traffic per token per stage boundary for activations of width d stored in b bytes ($b=2$ for bf16). Pipeline-parallel backpropagation ships activations forward and gradients backward ($2bd$ total) and is update-locked by the backward dependency. Local learning deletes the backward direction: bd per token per boundary, half the traffic, with no update locking. For the 400M class ($d=1,024$, bf16), this is 2 KiB versus 4 KiB per token per boundary. Subspace compression [Ramasinghe et al., 2025, Janson et al., 2026] is complementary: it projects boundary traffic to a rank $r \ll d$, and Ramasinghe et al. [2025] report the backward direction as the harder one to compress due to error accumulation; local learning removes exactly that direction, leaving only the

Table 5: Analytical inter-stage communication per token per stage boundary (d : model width; b : bytes per element, 2 for bf16; $r \ll d$: compression rank).

Scheme	Forward	Backward	Total	Update locking
Pipeline Parallelism (backprop)	bd	bd	$2bd$	locked
+ Subspace Compression [Ramasinghe et al., 2025]	br	br	$2br$	locked
Local Learning	bd	0	bd	none
+ Subspace Compression (fwd only)	br	0	br	none

easier forward one to compress. Finally, the auxiliary objective itself adds no traffic: MTP targets are the token stream every worker already holds.

F Training-compute overhead of the auxiliary choices

Table 6 gives the analytical training FLOPs per token of each auxiliary head, relative to end-to-end training of the same model. We count 6 FLOPs per token for every weight in a linear map (forward and backward) and add the causal attention term at sequence length 4,096. The softmax and the embedding lookup are not counted. Auxiliary heads exist only during training, so inference cost is unchanged, and the overhead below is what a training worker pays for hosting one auxiliary block.

Three points follow from the table. First, most of the overhead is the vocabulary projection that every local head needs to produce a next-token loss: 16.0% of end-to-end FLOPs per head at 125M and 7.1% at 400M. Second, the auxiliary designs add to that term as follows. Prepending a Transformer block (Transformer-aux over MLP-aux) adds 7.0 points per head at 125M and 3.9 at 400M. The FFN and norm of the MTP branch add 3.1 and 1.9 points, but predicting a second offset requires a second pass through the shared vocabulary projection, which adds another 16.0 and 7.1 points and is the main added cost of an MTP head. First-block MTP pays it at one head only. Compute alone does not order the results: all-block MTP costs the most and is worse than Transformer-aux at $K=4$ (Table 2).

Third, the vocabulary size is fixed, so the projection grows only with the model width while end-to-end cost triples. Every overhead therefore roughly halves from 125M to 400M. The total grows linearly with the number of auxiliary blocks, $K-1$. At 125M with $K=4$, a Transformer-aux head (0.24 GFLOP/token) costs more than the three-layer block it trains (0.19 GFLOP/token), whereas at 400M the block (0.64) costs more than the head (0.35).

All quality comparisons in this paper hold the token budget fixed, so the gaps in Table 1 are token-matched and do not charge local learning for the overheads in Table 6. Whether the overhead is recovered in wall-clock time depends on the deployment, because local updates also remove the pipeline bubbles and backward synchronization of pipelined backpropagation. Laskin et al. [2021] report net throughput gains for ResNets on ImageNet on real hardware despite the auxiliary computation. We do not measure throughput.

Table 6: Analytical training FLOPs of the auxiliary heads as a percentage of end-to-end training FLOPs per token at sequence length 4,096. End-to-end training costs 0.92 and 2.75 GFLOP/token at 125M and 400M. At $K=2$ there is one auxiliary block, so the total equals the per-head value. At $K=4$ the total sums over three auxiliary blocks, and first-block MTP places an MTP head at the first and Transformer-aux heads at the other two. Totals are computed before rounding.

	125M-class	400M-class
<i>Per auxiliary head</i>		
vocabulary projection alone	16.0%	7.1%
MLP-aux	19.1%	9.0%
Transformer-aux	26.1%	12.9%
MTP head	45.2%	21.9%
<i>Total, $K=4$ (three auxiliary blocks)</i>		
MLP-aux	57.2%	27.1%
Transformer-aux	78.2%	38.7%
Transformer-aux + MTP (first-block MTP)	97.3%	47.7%
all-block MTP	135.5%	65.8%

G MTP placement and weighting ablation

Table 7 compares the three MTP weightings defined in Appendix B at $K=4$, with Transformer-aux as the NTP-only reference. Placement is the lever that matters: moving the MTP pressure from all boundaries to the first one improves validation NLL by 0.008 (125M) and 0.024 (400M) nats, whereas the schedule within all-block MTP (linear vs. uniform) changes results by ~ 0.003 nats, at the level of run-to-run noise. The ablation is $K=4$ -only by construction: at $K=2$ there is a single auxiliary block, where first-block and all-block placement are the same thing and the linear and uniform schedules assign identical weights (1/2 per offset). The two $K=2$ runs are therefore identical configurations differing only in data-order noise; their gap (+0.017 vs. +0.020 at 125M) is our measured run-to-run noise floor of ~ 0.0025 nats.

Table 7: MTP placement/weighting ablation at $K=4$: Δ validation NLL vs. end-to-end (nats/token). The all-block uniform configuration was not run at 400M.

Configuration	125M-class	400M-class
Transformer-aux (NTP only, reference)	+0.078	+0.075
first-block MTP	+0.074	+0.062
all-block MTP, linear ramp	+0.082	+0.086
all-block MTP, uniform	+0.085	—

H Feature-space contrastive auxiliaries are shortcut-prone

Beyond token-space objectives, we tested feature-space contrastive auxiliaries in the Greedy InfoMax/CPC style [Löwe et al., 2019, van den Oord et al., 2018]: an InfoNCE loss over the block’s own future hidden states, with a Transformer-based auxiliary path and cosine-normalized scores. Two successive generations of shortcut defeated it, and we report both as observations.

Positional shortcut. With same-sequence negatives (pure CPC, 125M, $K=2$), the InfoNCE task is solvable from position information alone: Transformer residual streams carry absolute position, unlike the convolutional encoders of the CPC literature. Blocks collapsed into content-free position encoders; the auxiliary InfoNCE reached as low as ~ 0.09 nats (chance 8.3) and was ~ 0.28 at the stopping checkpoint, while the final-head loss stalled at ~ 7.25 , against ~ 2.9 for NTP-trained auxiliaries.

Document-identity shortcut. After fixing the above with same-position, cross-sequence negatives (which lower-bounds pure-position solutions at $\log B$ for batch size B), a hybrid NTP+CPC auxiliary (125M, $K=2$) still drove InfoNCE to ~ 0.5 nats, below the $\log B \approx 2.8$ bound, by combining position with coarse document fingerprints: position narrows the candidates to the target column, and the document fingerprint picks the block’s own sequence, so future-specific content is never required. The cross-entropy anchor prevented collapse, but the shortcut-directed pressure taxed the forwarded representation: the gap reached +0.154 nats at a single cut, worse than MLP-aux with three cuts (+0.132), against +0.028 for Transformer-aux and +0.017 for the MTP auxiliary at the same cut.

Table 8: Feature-space contrastive auxiliaries at 125M, $K=2$ (one cut), against the token-space auxiliaries of Table 1. The pure-CPC run collapsed into a position encoder and was stopped at step 4,000 of 5,100; its auxiliary InfoNCE (~ 0.28 nats at the stop) sits far below the same-sequence chance level while the final head never learns.

Auxiliary	val NLL	Δ vs. end-to-end (2.771)
CPC only	7.33	collapsed
NTP + CPC hybrid	2.925	+0.154
MLP-aux	2.816	+0.046
Transformer-aux	2.799	+0.028
Transformer-aux + MTP	2.788	+0.017

The common thread in both failures is that a language model’s hidden states are saturated with identity information (where a token sits, and which document it came from), and InfoNCE will

exploit whichever of these signals the negative set fails to control. Neither shortcut involves predicting anything about the future; the contrastive task is simply easier to win by other means. A negative-sampling scheme that suppresses both signals at once might still make contrastive auxiliaries viable for language-model pretraining, but designing one is beyond the scope of this study. Table 8 places both failures alongside the token-space auxiliaries at the same single cut.