TIDAL: ACCURACY RECOVERY IN LOCAL LEARNING WITH LIMITED FULL BACKPROPAGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Local learning trains consecutive groups of layers, called stages, with separate prediction losses. Removing backward dependencies between stages can increase distributed training throughput, but accuracy can fall sharply when each stage contains few layers. We introduce Tidal, a meta-algorithm that interleaves local learning with full backpropagation (BP), keeping the local objective and total number of training updates fixed. In four-stage, 1B-parameter language-model pretraining, one schedule uses full BP for 54.3% of updates and brings four local methods within 0.6 points of the full-BP reference in mean accuracy across nine tasks. With FluidPipe as the local learning method, a sparse schedule recovers approximately 86% of its accuracy gap using full BP for only 6.7% of updates. It exceeds a conventional BP run trained for the same number of BP updates by 8.8 accuracy points. Limited joint training thus enables existing local methods to retain high task accuracy under finer model partitions.

1 INTRODUCTION

Local learning can reduce the cost of distributed language-model pretraining by removing backward dependencies between parts of the model. In pipeline parallelism, consecutive groups of layers, called *stages*, reside on different accelerators (Huang et al., 2019; Narayanan et al., 2019). Full backpropagation (BP) trains all stages from the model's final prediction loss, so an earlier stage must wait for gradients from later stages. Local learning instead gives each stage its own prediction objective (Belilovsky et al., 2020; Laskin et al., 2020). For language-model pretraining, an auxiliary predictor attached to each nonfinal stage predicts the next token from that stage's output. Its prediction loss updates the stage and predictor without sending gradients to earlier stages. FluidPipe demonstrates that removing these backward dependencies can increase throughput over pipeline-parallel BP (Anonymous, 2026). At a fixed accelerator count, that throughput gain reduces the GPU-hours required to process a given amount of data.

Retaining accuracy becomes difficult as the model is divided more finely. Distributing a model across more accelerators can require fewer layers per stage. FluidPipe finds that such partitions sharply reduce accuracy across several local learning methods (Anonymous, 2026). This limits the use of local learning precisely when the model must be divided into more stages. We ask how much full BP is needed to recover this loss, and whether most of pretraining can still use local updates.

We introduce Tidal, a *meta-algorithm* that applies a schedule of local and full-BP updates to an existing local learning method. Local phases use that method's objectives and auxiliary predictors without modification. BP phases train the complete model jointly from the final prediction loss. The schedule can place BP at the beginning, intersperse it with local learning, or reserve it for the end. Each component is optional, allowing us to test whether recovery requires joint training before, during, or after substantial local learning.

Combining local and joint training has precedents: greedy layerwise pretraining ends with joint fine-tuning (Bengio et al., 2007), and Periodically Guided local Learning (PGL) alternates local and global training in vision (Bhatti & Moon, 2022). We examine this combination in billion-parameter language-model pretraining, focusing on the fine partitions where existing local methods lose accuracy.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

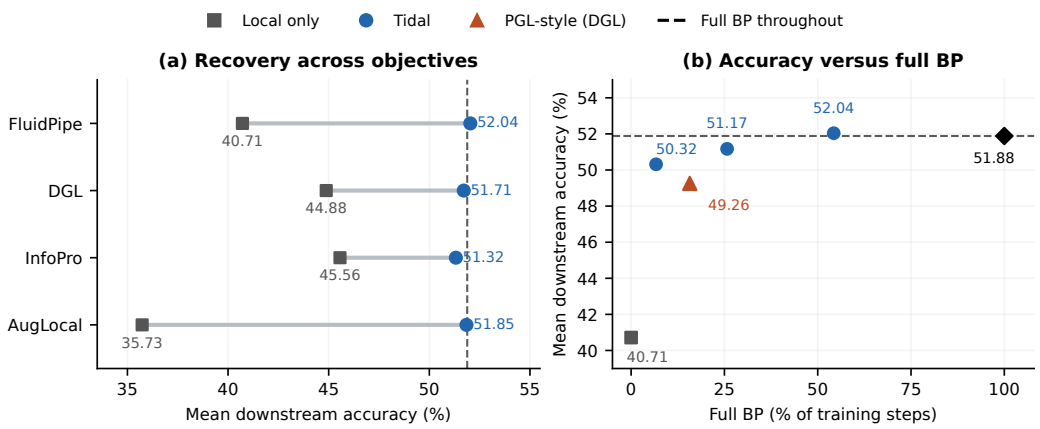


Figure 1: **Limited full BP recovers accuracy across local learning methods.** A four-stage, 1B language model is trained for 17,500 updates. (a) The same schedule, using 9,500 full-BP updates, brings all four local methods within 0.6 accuracy points of full BP. (b) Tidal applied to FluidPipe recovers approximately 86% of its accuracy gap with 1,170 full-BP updates (6.69%). Each point uses a different complete schedule, so BP amount and allocation both vary. The triangle is a PGL-style adaptation with DGL auxiliary predictors; its comparison with Tidal concerns complete recipes with different auxiliaries and schedules. Dashed lines mark full BP throughout. The conventional BP reference uses wider feed-forward layers (Section 4). Each point is one training run.

We apply Tidal to FluidPipe and to language-model adaptations of three prior local learning methods: DGL (Belilovsky et al., 2020), InfoPro (Wang et al., 2021), and AugLocal (Ma et al., 2024). In four-stage, 1B pretraining, one schedule using 54.3% BP brings all four within 0.6 points of the full-BP reference in mean accuracy across nine tasks (Figure 1). With FluidPipe, a sparse schedule recovers approximately 86% of the gap using only 6.7% BP. Recovery also extends to uneven partitions, including a two-layer first stage where purely local learning performs poorly.

The additional local updates contribute to this recovery. Sparse Tidal outperforms conventional BP trained for the same number of BP updates, and its accuracy increases during intervals containing only local learning. A local objective can therefore support most of pretraining even when using it on every update performs poorly.

2 Related Work

Combining local learning and joint training. Greedy layerwise pretraining learns a sequence of representations before joint fine-tuning (Bengio et al., 2007). Periodically Guided local Learning (PGL) alternates local learning and global training in vision, including networks with many locally trained groups (Bhatti & Moon, 2022). During its global phases, the main network is trained with the final prediction loss while auxiliary predictors continue to receive local losses. Interlocking backpropagation trains overlapping groups of layers, extending how far gradients travel across the network (Gomez et al., 2020). Tidal's full-BP phases optimize the final task loss and retain auxiliary parameters for later local learning. We test how far a common schedule can recover four local language-model methods under finer stage partitions, and use fixed-budget comparisons to study BP placement.

Local objectives and auxiliary predictors. DGL trains groups of layers with local supervised objectives (Belilovsky et al., 2020). InfoPro designs objectives to retain information useful to subsequent processing (Wang et al., 2021), while AugLocal increases the capacity of auxiliary predictors (Ma et al., 2024). FluidPipe studies auxiliary next-token predictors for partitioned language-model pretraining and documents the quality loss when stages contain few layers (Anonymous, 2026). These approaches determine the objectives and auxiliary predictors used during local learning. Tidal retains those choices and adds joint updates, testing whether the same schedule can recover accuracy across different local methods.

Distributed training schedules. GPipe, PipeDream, and zero-bubble pipeline schedules organize forward and backward work across devices (Huang et al., 2019; Narayanan et al., 2019; Qi et al., 2024). Our language-model implementation uses pipeline execution during full BP. Tidal determines which training updates use that execution and which use local learning. Local SGD and DiLoCo reduce communication between workers that each train a complete model (Stich, 2019; Douillard et al., 2023). Here, local learning refers to independently supervised stages within one model.

3 TIDAL: SCHEDULING LOCAL LEARNING AND FULL BACKPROPAGATION

Given a network divided into stages, a local learning method and a schedule, Tidal selects either local learning or full BP at each training update. Local updates train each stage with the chosen method's objective. Full-BP updates train all stages jointly from the loss at the network's final output. When local learning resumes, it uses the same objectives and auxiliary predictors.

Local and joint updates. Let K be the number of stages and x the network input. Starting from $h_0 = x$, stage i computes $h_i = f_i(h_{i-1}; \theta_i)$, where f_i is its group of layers, θ_i its trainable parameters, and h_i its output representation. A learned final predictor c maps h_K to the task prediction. For target y and prediction loss ℓ, the full-network loss is $\mathcal{L}_{\text{full}} = \ell(c(h_K), y)$. Full BP differentiates this loss through every stage.

In local learning, each nonfinal stage has an auxiliary predictor $a_i(\cdot; \phi_i)$ with trainable parameters ϕ_i. Its local objective $\mathcal{L}_i$ trains that stage and its predictor. For next-token prediction, the prediction term is $\ell(a_i(h_i; \phi_i), y)$. A method can also include other terms, such as InfoPro's reconstruction objective (Wang et al., 2021). The input h_{i-1} is held fixed during differentiation, so $\mathcal{L}_i$ updates θ_i and ϕ_i without updating earlier stages. The final stage uses $\mathcal{L}_K = \mathcal{L}_{\text{full}}$, with the same restriction on gradients to earlier stages.

Let $u_t = 1$ select full BP at update t, and $u_t = 0$ select local learning. The gradient $g_{i,t}$ supplied to the optimizer for stage i is

$$g_{i,t} = \begin{cases} \nabla_{\theta_i} \mathcal{L}_{\text{full}}, & u_t = 1, \\ \nabla_{\theta_i} \mathcal{L}_i, & u_t = 0. \end{cases} \tag{1}$$

Here, ∇_{θ_i} denotes differentiation with respect to that stage's parameters. The final predictor receives task-loss gradients in both modes. Auxiliary parameters receive updates only during local learning and are retained during BP. Model parameters, optimizer state, and the learning-rate schedule continue across switches. The learning rate is indexed by the total number of updates, including both modes. At evaluation, inputs pass through the complete network to its final task predictor, and auxiliary predictors are omitted.

Amount and allocation of BP. A schedule may use BP at the beginning of training, between local intervals, and at the end. We call these its opening, intermediate, and closing BP phases. A short intermediate BP phase is a *pulse*. For T total updates, the BP budget and fraction are

$$B = \sum_{t=1}^{T} u_t = O + I + C, \qquad \rho = B/T, \tag{2}$$

where O, I, and C count opening, intermediate, and closing BP updates. The remaining $T - B$ updates use local learning. Any of O, I, or C can be zero, so Tidal also includes opening-only and closing-only BP schedules.

At fixed B, moving BP updates to one part of training requires removing them from another. Such comparisons test placement without changing how much joint training the model receives. The same intermediate budget I can also be divided into many short phases or fewer long phases. All schedules in the main study are fixed before training begins.

Training cost. FluidPipe measures higher throughput for local learning than pipeline-parallel BP in its evaluated configurations (Anonymous, 2026). For a common batch size and fixed GPU allocation, let t_{local} and t_{BP} be the mean update times, including each mode's computation and communication. The local time includes the auxiliary predictors. Relative to T full-BP updates, the

approximate time saved by Tidal is

$$\Delta\mathcal{T} = (T - B)(t_{\mathrm{BP}} - t_{\mathrm{local}}) - \mathcal{T}_{\mathrm{switch}}, \tag{3}$$

where $\mathcal{T}_{\mathrm{switch}}$ is the total switching overhead. With negligible switching overhead, replacing BP with faster local updates reduces training time and, at a fixed GPU count, GPU-hours. We use the BP fraction ρ as the experimental budget because update times depend on the local method, hardware, and parallel configuration. This accounting compares runs with the same total number of updates. We also compare with shorter conventional BP training to test whether the additional local updates provide useful learning beyond the BP budget alone.

4 Recovery in 1B language-model pretraining

The experiments test recovery across local objectives and partitions, then examine the contribution of local updates and whether recovery survives a return to local learning. We pretrain a 1B-parameter, 16-layer decoder on FineWeb-Edu for 17,500 updates, processing 36.7 billion tokens (Penedo et al., 2024). The principal comparison uses four stages of four layers each. Two-stage and uneven partitions test whether recovery extends to other divisions of the same model. Our primary metric is mean zero-shot accuracy across nine tasks, measured with `lm-evaluation-harness` (Gao et al., 2021). Appendices A and B provide the protocols, exact schedules and supporting results.

Most recipes have one training run, with a second seed for two allocations. The 17,500-update Tidal and BP runs share the data budget, AdamW optimizer and learning-rate schedule, with peak rate 5×10^{-4}. The conventional BP reference uses 11,008 hidden units in each feed-forward layer, versus 8,192 for Tidal and the local methods. The other decoder dimensions are unchanged. The four local-method references use 6×10^{-4}, so their recovery comparisons concern the evaluated recipes. The uneven-partition comparisons match the rate at 6×10^{-4} for both local learning and Tidal.

4.1 One schedule recovers four local learning methods

Local learning with four stages leaves substantial accuracy gaps across FluidPipe and language-model adaptations of DGL, InfoPro and AugLocal. We apply the same Tidal schedule to all four, retaining each method's local objective and auxiliary predictors. The schedule alternates 2,000 BP updates and 2,000 local updates four times, then finishes with 1,500 BP updates. Its 9,500 BP updates constitute 54.29% of training, leaving 8,000 local updates.

All four methods finish within 0.6 accuracy points of the full-BP reference (Figure 1a). Gains over their local references range from 5.76 to 16.11 points, and each method improves on all nine task metrics. The largest recovery occurs for AugLocal, which improves from 35.73% under local learning to 51.85% with Tidal. Its final accuracy is close to full BP at 51.88%. The same BP schedule is effective across local objectives with very different accuracies when used alone.

4.2 Substantial recovery with few BP updates

With four-stage FluidPipe, a sparse schedule reaches 50.32%, compared with 40.71% for local learning and 51.88% for full BP (Figure 1b). It uses only 1,170 BP updates, or 6.69% of training, recovering 85.96% of the accuracy gap. Recovery is the improvement over local learning divided by the full-BP-minus-local accuracy difference. A schedule using 25.71% BP reaches 51.17%. The sparse result leaves a 1.57-point gap to full BP despite performing more than 93% of its updates locally.

The sparse schedule begins with 1,000 BP updates, then repeatedly performs 990 local updates and ten BP updates, with a shorter final local interval. Its last ten updates use BP. Most of its joint training is therefore concentrated at the beginning. The complete schedule establishes recovery with little BP, while the contribution of its later pulses is not isolated by this comparison. Because the larger-budget schedules also allocate BP differently, Figure 1b compares complete schedules rather than varying BP amount alone.

Equal and uneven partitions. For equal partitions, FluidPipe's local references reach 49.65% at two stages and 40.71% at four. Applying the same sparse schedule gives 51.75% and 50.32%,

Table 1: **Recovery under uneven 1B partitions.** Layer counts are listed in input-to-output order. Every run uses 17,500 updates; Tidal and its local reference share the peak learning rate 6×10^{-4} and decay schedule. One training run per recipe.

		Mean accuracy (%)		
Partition	BP updates	Local	Tidal	Gain
8/3/3/2	1,170	46.83	51.54	+4.71
2/14	9,500	35.76	52.79	+17.02

respectively. The common 9,500-BP schedule reaches 51.84% and 52.04%. The sparse schedule leaves a larger residual gap at four stages, while the larger BP budget restores near-reference task accuracy at both (Figure 3b).

Recovery also extends to uneven partitions (Table 1). With stages of 8, 3, 3 and 2 layers, the sparse schedule improves accuracy by 4.71 points. With a two-layer first stage and a fourteen-layer second stage, local learning reaches only 35.76%. The common 9,500-BP schedule raises this to 52.79%, an improvement of 17.02 points. Both comparisons hold the partition and learning rate fixed between local learning and Tidal. Recovery thus extends beyond equal partitions, including a case where even two stages leave a severe local-learning deficit.

Comparison with a PGL-style recipe. The PGL-style language-model adaptation uses DGL auxiliary predictors and approximately 15.71% BP. At two stages, it reaches 44.01%, compared with 51.66% for Tidal with FluidPipe using 14.98% BP, the closest available Tidal budget. At four stages, it reaches 49.26%, compared with 50.32% for sparse Tidal using 6.69% BP. Tidal attains higher accuracy with less BP in both comparisons, although the different auxiliary networks and allocations prevent attributing the difference to the schedule alone. Implementation details are in Appendix B.3.

4.3 Local updates contribute to recovery

To test what the local updates add beyond BP, we train the conventional BP reference from scratch for exactly 1,170 updates, matching sparse Tidal's BP count. Warmup and cosine decay are rescaled to complete within that horizon. This shorter run reaches 41.50% mean accuracy, compared with 50.32% for sparse Tidal: an advantage of 8.82 points, with Tidal ahead on all nine task metrics. The BP count alone is therefore insufficient to account for sparse Tidal's final accuracy in this comparison. Tidal's additional local updates process more data; the comparison holds the BP count fixed, not total computation.

The trajectory also shows accuracy improving during local learning itself (Figure 2). Panel (b) indexes progress by cumulative BP updates, making progress during local learning visible as vertical segments. The first 500 local updates increase accuracy from 38.43% to 39.59%, a gain of about 1.2 points. After the next BP pulse, another 500 local updates raise accuracy from 42.21% to 43.86%. Across the sixteen observed 500-update intervals containing only local learning, accuracy increases in twelve, with smaller and less consistent changes late in training. At this partition, local updates can therefore improve accuracy after joint training even though using local learning throughout leaves a substantial deficit.

4.4 A BP opening can support prolonged local learning

If BP is needed only to prepare the model for local learning, a single opening phase may suffice. At two stages, 2,000 initial BP updates followed by 15,500 local updates reach 51.79% accuracy, compared with 49.65% for purely local learning and 51.88% for full BP. A BP opening is therefore sufficient for near-reference task accuracy in this two-stage configuration, with 88.6% of pretraining performed locally and no later BP.

The reverse phase order allocates 2,000 updates to local learning followed by 15,500 BP updates. It reaches 52.41% accuracy at the end of training. The initially local model can thus reach high accuracy through subsequent joint training. The two directions use very different BP budgets, and the local-to-BP continuation starts a fresh optimizer, so their final accuracies do not rank the two allocations.

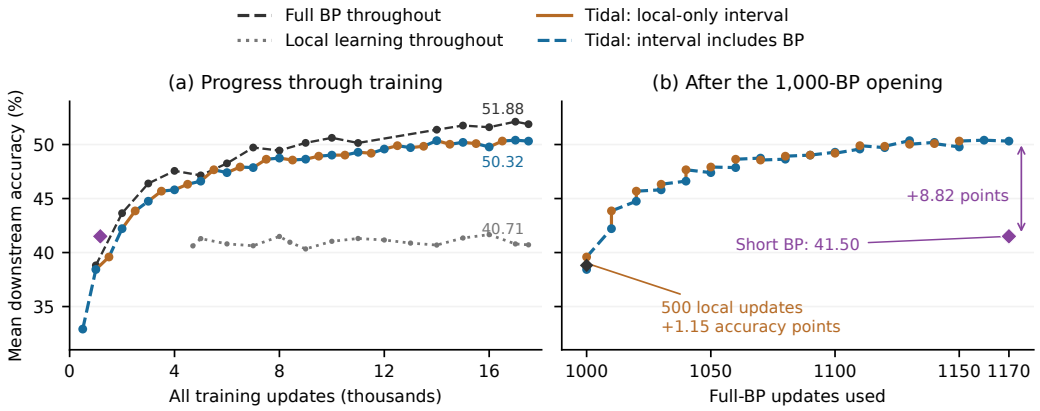


Figure 2: **Sparse Tidal improves on BP alone at the same BP count.** Tidal uses 1,170 BP updates across 17,500 total updates in four-stage FluidPipe. (a) Accuracy through training, compared with conventional BP and purely local learning. (b) The same Tidal observations after the 1,000-BP opening, indexed by cumulative BP updates. After the opening, orange segments contain 500 local updates and no BP; the intervening dashed segments contain 490 local updates and ten BP updates. Each curve is one run, with evaluations every 500 updates for Tidal. The BP and local references have fewer retained evaluations, which are connected by straight lines. Purple diamonds show the separate 1,170-update BP run, whose warmup and decay are rescaled to its shorter horizon. It reaches 41.50%, compared with 50.32% for Tidal. The longer BP trajectory contributes only its 1,000-update observation to panel (b). Neither axis measures elapsed time.

4.5 Recovery persists without a final BP phase

We next test whether four-stage recovery requires BP at the end of training, holding its total BP budget fixed. Each schedule uses 7,000 BP updates within the 17,500-update horizon. They concentrate more BP early, distribute it uniformly, or place more of it late (Figure 3a). The first two end with local learning, and the late allocation ends with BP.

Early and uniform allocation reach 51.50% and 51.73% in the first run, and 51.06% and 51.04% in the second. The late allocation reaches 50.81% in its one run. The ordering between early and uniform allocation changes across runs. Both retain substantial recovery after returning to local learning, showing that a closing BP phase is not required in these four-stage schedules.

High final accuracy does not imply that every local phase improves the model. In the uniform schedule's second run, the final 1,750 local updates reduce accuracy from 51.65% to 51.04%. Most of the recovered accuracy remains, although this local phase does not improve it.

4.6 Scope of the recovery result

We use smaller vision settings to test whether Tidal both recovers accuracy lost under local learning and improves on conventional BP trained for the same BP budget. These settings allow repeated schedule comparisons at lower cost than 1B pretraining.[1]

Tidal improves on local learning in ViT fine-tuning on CIFAR-100 and CINIC-10, and in ResNet-101 training from scratch on CIFAR-100. In all three settings, however, shorter conventional BP attains higher mean accuracy. These settings reproduce the local-learning accuracy deficit and its recovery, but not the advantage over shorter BP observed in 1B pretraining. Appendix C reports these comparisons and their protocols.

[1]Applying FluidPipe's measured throughput of 92,242 tokens/s on four H100 GPUs to our 36.7-billion-token horizon gives approximately 440 GPU-hours for full BP, excluding evaluation, checkpointing and restarts (Anonymous, 2026).

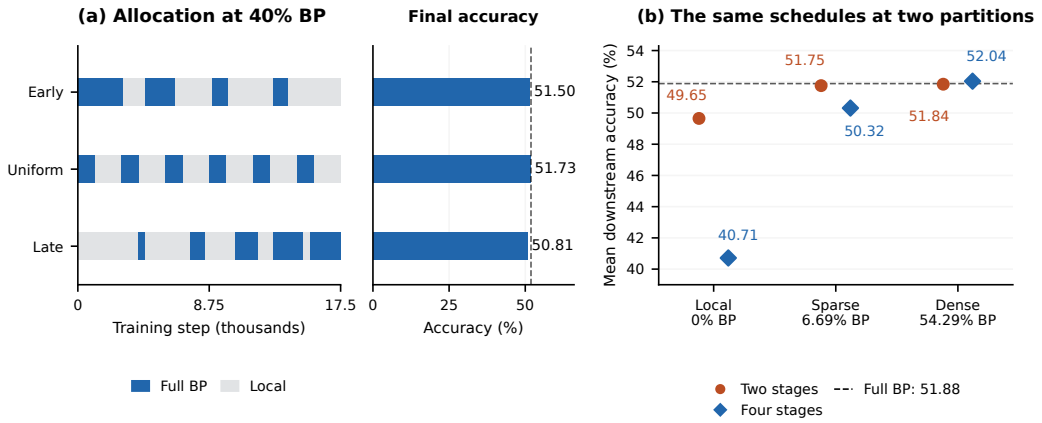


Figure 3: **Recovery under different BP placements and stage counts.** All results follow 17,500 updates. (a) Three four-stage schedules allocate the same 7,000 BP updates, or 40% of training. Colored segments mark BP, and gray segments mark local learning. Bars show final accuracy for seed 42. The text also reports the second runs for early and uniform allocation. Early and uniform allocation both finish locally. (b) The same sparse and larger-budget schedules are used at two and four stages. The larger budget restores near-reference accuracy at both partitions. Dashed lines mark the full-BP reference.

5 Discussion

Local objectives remain useful under finer partitions. A local objective that performs poorly throughout pretraining can still supply useful updates between BP phases. Tidal's recovery across objectives and partitions supports retaining these local methods under finer partitions, with joint training confined to selected phases.

Joint training can prepare a model for further local learning. The gains during local intervals suggest that BP can improve the model's ability to learn from local objectives. One possible explanation is that joint updates make each stage's representations more useful to later stages. BP also changes optimizer state, and auxiliary predictors adapt when local learning resumes. The current experiments do not separate these effects. Continuing from the same checkpoint while controlling which weights and optimizer states change would help identify what enables subsequent local learning.

Choosing when to resume joint training. Future schedules could use the progress made during local learning to decide when to return to BP. This requires testing whether a slowdown in local learning predicts a benefit from joint updates, measured both immediately and after local learning resumes. Varying pulse spacing at a fixed BP budget would test that prediction and help distinguish useful recurrence from unnecessary switches.

6 Conclusion

Tidal combines existing local learning objectives with scheduled full-BP updates. In 1B pretraining, one schedule brings four local methods within 0.6 accuracy points of the BP reference, and recovery extends to finer equal and uneven partitions. Sparse Tidal recovers approximately 86% of FluidPipe's accuracy gap with only 6.7% BP and exceeds a conventional run restricted to the same BP count by 8.82 points. Accuracy gains during local phases further show that local learning contributes to the recovered model. The result is a way to retain local learning for most of pretraining under partitions where using it alone gives poor accuracy.

Reproducibility statement

The appendix provides training protocols, exact schedules, task scores and individual-run results. The accompanying source manifest and analysis scripts reproduce the numerical results and figures.

AI use statement

Generative AI was used to aid and polish the writing and to assist with literature retrieval and discovery.

References

Anonymous. FluidPipe: Local learning for LLM pretraining and its limits. Under submission, 2026. Authors redacted due to double blindness.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Decoupled greedy learning of CNNs. In *International Conference on Machine Learning (ICML)*, pp. 736–745. PMLR, 2020. doi: 10.48550/arxiv.1901.08164. URL https://doi.org/10.48550/arxiv.1901.08164.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy Layer-Wise Training of Deep Networks. In *Advances in Neural Information Processing Systems 19*, 2007. doi: 10.7551/mitpress/7503.003.0024. URL https://doi.org/10.7551/mitpress/7503.003.0024.

Hasnain Irshad Bhatti and Jaekyun Moon. Locally supervised learning with periodic global guidance. In *ICML Workshop on Hardware Aware Efficient Training*, 2022. URL https://arxiv.org/pdf/2208.00821. arXiv:2208.00821.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020. doi: 10.1609/aaai.v34i05.6239. URL https://doi.org/10.1609/aaai.v34i05.6239.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019. doi: 10.18653/v1/n19-1300. URL https://doi.org/10.18653/v1/n19-1300.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Luke N. Darlow, Elliot J. Crowley, Antreas Antoniou, and Amos J. Storkey. CINIC-10 is not ImageNet or CIFAR-10. *arXiv preprint arXiv:1810.03505*, 2018. URL https://arxiv.org/abs/1810.03505.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

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

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation. Zenodo, software release v0.0.1, 2021. URL https://doi.org/10.5281/zenodo.5371629.

Aidan N Gomez, Oscar Key, Kuba Perlin, Stephen Gou, Nick Frosst, Jeff Dean, and Yarin Gal. Interlocking backpropagation: Improving depthwise model-parallelism. *arXiv preprint arXiv:2010.04116*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. doi: 10.1109/cvpr.2016.90. URL https://doi.org/10.1109/cvpr.2016.90.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. GPipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019. doi: 10.48550/arxiv.1811.06965. URL https://doi.org/10.48550/arxiv.1811.06965.

Michael Laskin, Luke Metz, Seth Nabarro, Mark Saroufim, Badreddine Noune, Carlo Luschi, Jascha Sohl-Dickstein, and Pieter Abbeel. Parallel training of deep networks with local updates. *arXiv preprint arXiv:2012.03837*, 2020. URL https://arxiv.org/abs/2012.03837.

Chenxiang Ma, Jibin Wu, Chenyang Si, and Kay Chen Tan. Scaling supervised local learning with augmented auxiliary networks. In *International Conference on Learning Representations (ICLR)*, 2024. doi: 10.48550/arxiv.2402.17318. URL https://doi.org/10.48550/arxiv.2402.17318.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018. doi: 10.18653/v1/d18-1260. URL https://doi.org/10.18653/v1/d18-1260.

Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R Devanur, Gregory R Ganger, Phillip B Gibbons, and Matei Zaharia. Pipedream: Generalized pipeline parallelism for dnn training. In *Proceedings of the 27th ACM symposium on operating systems principles*, pp. 1–15, 2019. doi: 10.1145/3341301.3359646. URL https://doi.org/10.1145/3341301.3359646.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016. doi: 10.18653/v1/p16-1144. URL https://doi.org/10.18653/v1/p16-1144.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*, 2024.

Penghui Qi, Xinyi Wan, Guangxing Huang, and Min Lin. Zero bubble (almost) pipeline parallelism. In *International Conference on Learning Representations (ICLR)*, 2024. doi: 10.48550/arxiv.2401.10241. URL https://openreview.net/forum?id=tuzTN0eIO5.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020. doi: 10.1609/aaai.v34i05.6399. URL https://doi.org/10.1609/aaai.v34i05.6399.

Sebastian U Stich. Local SGD converges fast and communicates little. In *International Conference on Learning Representations (ICLR)*, 2019. doi: 10.48550/arxiv.1805.09767. URL https://doi.org/10.48550/arxiv.1805.09767.

Yulin Wang, Zanlin Ni, Shiji Song, Le Yang, and Gao Huang. Revisiting locally supervised learning: an alternative to end-to-end training. In *International Conference on Learning Representations (ICLR)*, 2021. doi: 10.48550/arxiv.2101.10832. URL https://doi.org/10.48550/arxiv.2101.10832.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text (W-NUT @ EMNLP)*, 2017. doi: 10.18653/v1/w17-4413. URL https://doi.org/10.18653/v1/w17-4413.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019. doi: 10.18653/v1/p19-1472. URL https://doi.org/10.18653/v1/p19-1472.

A TRAINING AND EVALUATION PROTOCOLS

A.1 LANGUAGE-MODEL RECIPES

The local-learning and Tidal backbone has 16 decoder layers, hidden width 2,048, intermediate width 8,192, 32 attention heads, eight key/value heads and vocabulary size 128,256. The conventional BP reference has intermediate width 11,008; its other listed dimensions agree. These comparisons therefore concern the evaluated models rather than an architecture-matched BP ablation. The two-stage and four-stage partitions contain $[8, 8]$ and $[4, 4, 4, 4]$ decoder layers, respectively; auxiliary modules are additional to these layers. Training uses bfloat16, context length 2,048 and an accumulated batch of 1,024 sequences per update. All full-length runs process 36.7 billion FineWeb-Edu tokens over 17,500 updates.

Unless specified otherwise, Tidal and full BP use AdamW with peak learning rate 5×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 0.1 and gradient clipping at norm 1. The rate warms up for 1,000 updates, then decays by cosine over 16,500 updates to 5×10^{-5}. Four-stage local references use peak and final rates 6×10^{-4} and 6×10^{-5}; the two-stage FluidPipe reference uses the Tidal rates. The primary runs use training seed 42; two placement schedules also use seed 1337. Means across downstream tasks do not represent replication across training seeds.

The language-model adaptations retain distinct auxiliary predictors and objectives. FluidPipe adds a decoder block and next-token predictor to each nonfinal stage. DGL uses a classifier with hidden width 2,048 and no additional decoder block. InfoPro adds a reconstruction objective and a 512-dimensional projection, with information-term weight 0.1. AugLocal uses pyramidal auxiliary depth with at most three blocks. Backbone weights, Adam moments and the learning-rate clock continue across Tidal switches; auxiliary parameters are retained for later local phases. The local and pipeline execution paths use separate data loaders, so their example order is not matched.

A.2 SCHEDULES AND FULL-BP COUNTS

A phase list specifies consecutive phase lengths, starting with BP and alternating with local learning unless stated otherwise. A superscript repeats a pair; $\|$ concatenates lists. The principal schedules are

$$\begin{aligned} \text{Dense, } K = 2, 4 &: [2000, 2000]^4 \| [1500], && B = 9500, \\ \text{Sparse, } K = 2, 4 &: [1000] \| [990, 10]^{16} \| [490, 10], && B = 1170, \\ \text{Intermediate, } K = 4 &: [500, 1500]^8 \| [500, 1000], && B = 4500. \end{aligned}$$

Each list sums to 17,500 updates, and every reported endpoint uses the final model, including schedules that end locally. Of the sparse schedule's 1,170 BP updates, 1,000 occur in the opening. Its accuracy measures the complete allocation; the available single-switch runs do not isolate the contribution of the later pulses at that budget.

A.3 TASKS, SCORING AND REFERENCES

All nine tasks are evaluated zero-shot using the following fields:

Task	Metric
HellaSwag	`acc_norm`
ARC-Easy	`acc_norm`
ARC-Challenge	`acc_norm`
PIQA	`acc_norm`
WinoGrande	`acc`
OpenBookQA	`acc_norm`
SciQ	`acc_norm`
LAMBADA OpenAI	`acc`
BoolQ	`acc`

Task definitions follow the original benchmarks (Zellers et al., 2019; Clark et al., 2018; Bisk et al., 2020; Sakaguchi et al., 2020; Mihaylov et al., 2018; Welbl et al., 2017; Paperno et al., 2016;

Clark et al., 2019). Evaluation uses the Hugging Face backend of `lm-evaluation-harness`, bfloat16, SDPA attention and batch size 16. Both execution paths evaluate the complete backbone through its final normalization and language-model predictor, excluding auxiliary predictors. Means and differences are computed from unrounded task accuracies and reported in percent or percentage points.

For a scheduled run, recovery is

$$R = 100\,\frac{A_{\text{scheduled}} - A_{\text{local}}}{A_{\text{full}} - A_{\text{local}}}, \tag{4}$$

where $A_{\text{scheduled}}$, A_{local}, and A_{full} are final accuracies for the scheduled, purely local, and full-BP runs, respectively. Accuracies are averaged across training seeds when multiple runs are available. The primary full-BP reference scores 51.88% at update 17,500; we use final rather than best-checkpoint scores throughout. A separate BP run with the narrower intermediate width 8,192 and peak rate 3×10^{-4} reaches 51.11%. A four-stage local reference at peak rate 5×10^{-4} reaches 39.39%; its rate is recorded in the experiment ledger, but the original launch configuration is unavailable. The reported recovery fractions retain the stronger local reference at 6×10^{-4} used in the four-method comparison.

B Supporting language-model results

B.1 Per-task recovery

Table 2 gives the task scores underlying Figure 1a. The common 9,500-BP schedule improves every task metric for all four local methods.

Table 2: **Final task accuracies in the four-stage 1B comparison.** L and T denote local learning throughout and Tidal, respectively. The mean is calculated before rounding individual task scores.

	Full BP	FluidPipe		DGL		InfoPro		AugLocal	
Task		L	T	L	T	L	T	L	T
HellaSwag	46.66	31.64	46.23	33.05	45.79	35.02	45.14	28.33	46.59
ARC-Easy	58.50	44.78	58.88	48.78	57.95	50.21	56.99	36.66	58.42
ARC-Challenge	32.42	22.70	32.51	24.49	33.02	28.07	30.89	23.55	31.57
PIQA	69.59	60.66	69.53	63.93	68.93	63.76	68.55	56.15	69.15
WinoGrande	52.72	50.99	53.28	51.07	52.25	49.57	52.49	51.70	52.17
OpenBookQA	33.40	30.00	36.00	30.00	37.00	28.40	35.80	29.00	35.80
SciQ	76.60	59.10	75.10	69.90	72.90	67.80	74.10	44.60	75.20
LAMBADA	39.10	16.38	38.19	24.01	38.29	27.34	37.01	1.09	36.52
BoolQ	57.95	50.18	58.62	58.68	59.27	59.88	60.92	50.52	61.19
Mean	51.88	40.71	52.04	44.88	51.71	45.56	51.32	35.73	51.85

B.2 Schedules at two and four stages

Table 3 collects the recurring FluidPipe schedules. All use the common optimizer and learning-rate recipe, but differ in BP amount and allocation. For the matched sparse and dense schedules across stage counts, model and token-batch settings agree; the partition and pipeline/tensor-parallel layout change. The local references score 49.65% at two stages and 40.71% at four, against 51.88% for full BP.

The equal-length schedules use $[10, 10]^{875}$, $[100, 100]^{87}\|[100]$ or the dense schedule above. The three local-heavy schedules begin with 1,000 BP updates, followed by repetitions of the listed local/BP pair. The six-cycle schedule is $[437, 2479]^5\|[437, 2483]$. The additional four-stage schedule is $[875, 2625]^5$. These outcomes show recovery across different allocations; their ordering does not isolate pulse frequency or BP amount.

Table 3: **Recurring FluidPipe schedules at a common 17,500-update horizon.** Each row is one training run with seed 42. Dense, sparse and intermediate schedules are defined in Appendix A.2.

Stages	Schedule	BP updates	BP (%)	Accuracy (%)
2	Sparse	1,170	6.69	51.75
2	1,400 local / 100 BP	2,100	12.00	51.94
2	1,000 local / 100 BP	2,500	14.29	52.11
2	Six cycles, short opening	2,622	14.98	51.66
2	1,250 local / 250 BP	3,750	21.43	51.82
2	10-step phases	8,750	50.00	51.63
2	100-step phases	8,800	50.29	51.99
2	Dense: 2,000-step phases	9,500	54.29	51.84
4	Sparse	1,170	6.69	50.32
4	875 BP / 2,625 local	4,375	25.00	50.78
4	Intermediate	4,500	25.71	51.17
4	Dense: 2,000-step phases	9,500	54.29	52.04

B.3 Comparison with a PGL-style schedule

The adaptation of periodic global guidance (Bhatti & Moon, 2022) repeats 1,250 local and 250 BP updates eleven times, then ends with 1,000 local updates. It uses DGL auxiliary networks and 2,750 BP updates over the same 17,500-update horizon (Table 4). Auxiliary parameters are retained but not updated during BP, whereas original PGL continues their local-loss updates during global training (Bhatti & Moon, 2022, Eq. 4). The original launch configurations are unavailable: schedule and auxiliary settings follow the implementation notes, while accuracy and training horizon are verified from evaluation records.

Table 4: **PGL-style and Tidal recipes after 17,500 updates.** Local references correspond to each auxiliary architecture. The BP fractions marked d follow the documented adaptation.

K	Local method	Schedule	BP (%)	Accuracy (%)
2	DGL	Local only	0.00	47.33
2	DGL	PGL-style	15.71^{d}	44.01
2	FluidPipe	Local only	0.00	49.65
2	FluidPipe	Tidal, six cycles	14.98	51.66
4	DGL	Local only	0.00	44.88
4	DGL	PGL-style	15.71^{d}	49.26
4	FluidPipe	Local only	0.00	40.71
4	FluidPipe	Sparse Tidal	6.69	50.32

The adaptation falls below its DGL local reference at two stages and exceeds it at four, where the local reference uses a higher learning rate. The low-budget FluidPipe schedules exceed the adaptation at both partitions. This compares complete recipes with different auxiliary networks and schedules; it does not isolate BP allocation or imply equal computational cost.

B.4 Accuracy through local and full-BP phases

The sparse trajectory in Figure 2 contains 35 evaluations, spaced 500 updates apart. Sixteen intervals contain only local learning: updates 1,000–1,500, 2,000–2,500, and so on through 16,000–16,500. Accuracy increases in twelve, including one increase of only 0.002 points. Each interval covers the first 500 updates of a 990-update local phase; the intervening observations mix 490 local updates with ten BP updates and cannot isolate a pulse's effect. These are changes within one training run.

The dense trajectory gives a different pattern (Figure 4). Its first 2,000 local updates improve accuracy by 2.77 points; the next two local phases change it by -0.47 and -0.57, and the last by $+0.08$. Thus, strong early progress does not imply sustained gains during every local phase. Comparisons by cumulative BP count retain the original learning-rate schedules and differ in total updates and data exposure. They describe training progress rather than an equal-time comparison or an isolated effect of the preceding local phase.

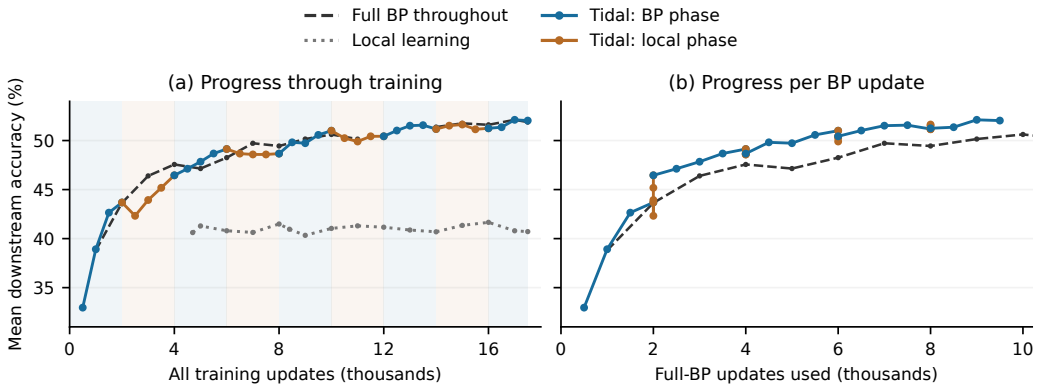


Figure 4: **The larger-budget schedule makes uneven progress during local learning.** Four-stage FluidPipe uses the dense schedule with 9,500 BP updates. Nine-task accuracy is indexed by total updates (left) and cumulative BP updates (right). Vertical orange segments contain local updates only. Each curve is one run; lines connect recorded evaluations, with gaps for unavailable observations. The BP reference has no evaluation at exactly 9,500 updates.

Figure 5 extends the phase-end comparison to all four local objectives under the same dense schedule. Each observation evaluates the complete model after a local or BP phase.

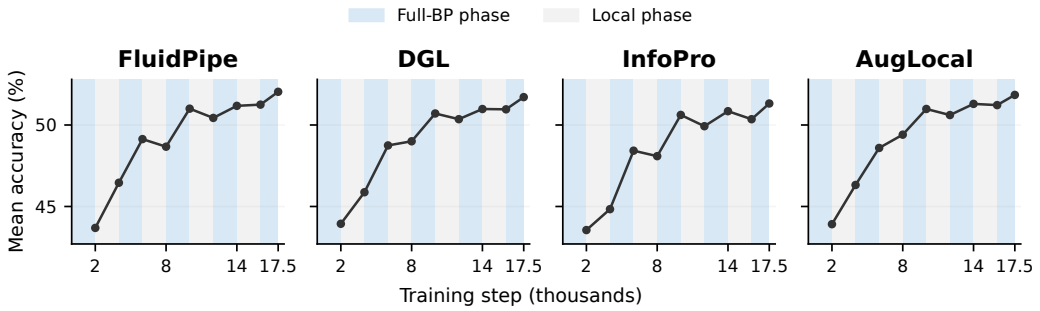


Figure 5: **Accuracy improves repeatedly during full-BP phases.** The four Tidal runs from Figure 1a, evaluated at each phase end. Background shading marks the training mode. Connecting lines do not resolve changes within a phase or identify which parameters produce the gains.

B.5 Comparison with shorter BP training

The short-BP control uses the conventional reference's model, data source, token batch, AdamW settings and initialization seed 42. It trains from scratch for 1,170 updates, processing 2.45 billion tokens. Warmup lasts 67 updates, followed by 1,103 updates of cosine decay from 5×10^{-4} to 5×10^{-5}. This rescales the full run's warmup/decay proportions to the shorter horizon. Evaluation follows Appendix A.3 with `lm-evaluation-harness` version 0.4.11. The sparse run's historical records do not retain exact evaluator and dataset revisions, so their equality cannot be verified retrospectively.

Sparse Tidal scores higher on every task, with an 8.82-point mean advantage (Table 5). The comparison holds BP count fixed; Tidal additionally performs 16,330 local updates and processes more training data.

B.6 Placement at a fixed BP budget

The three four-stage schedules in Table 6 allocate 7,000 BP updates within the same horizon. Early and uniform allocations start in BP mode and use $[3000, 1500, 2000, 2500, 1000, 3000, 1000, 3500]$

Table 5: **Sparse Tidal and shorter conventional BP at 1,170 BP updates.** Final task accuracy, one training run per recipe. Short BP completes its learning-rate decay within 1,170 updates.

Task	Short BP (%)	Sparse Tidal (%)	Difference (points)
HellaSwag	29.62	43.24	+13.62
ARC-Easy	42.89	56.86	+13.97
ARC-Challenge	24.66	32.25	+7.59
PIQA	59.68	67.74	+8.05
WinoGrande	51.38	51.78	+0.39
OpenBookQA	31.20	35.00	+3.80
SciQ	58.50	71.40	+12.90
LAMBADA	16.71	32.84	+16.13
BoolQ	58.87	61.74	+2.87
Mean	41.50	50.32	+8.82

and $[1167, 1750]^4 \| [1166, 1750]^2$, respectively. Both end locally. The late allocation starts locally, uses $[4000, 500, 3000, 1000, 2000, 1500, 1000, 2000, 500, 2000]$, and ends in BP. Early and uniform schedules have two training seeds; the late schedule has one, so its relative ranking is less well determined.

Table 6: **Placement of 7,000 BP updates in four-stage 1B pretraining.** Accuracy is measured after all 17,500 updates.

Allocation	Seed 42	Seed 1337
Early	51.50	51.06
Uniform	51.73	51.04
Late	50.81	—

B.7 SINGLE-SWITCH CONTINUATIONS

Table 7 compares the completed two-stage recipes, using the $[8, 8]$ partition during local learning. The first continuation performs 2,000 BP updates followed by 15,500 local updates, preserving the learning-rate clock through the switch. The reverse continuation performs 2,000 local updates followed by 15,500 BP updates, starting a fresh optimizer with the same nominal peak rate and cosine schedule.[2]

The longer local opening uses 7,000 local updates at peak rate 3×10^{-4}, then 10,500 BP updates with a fresh optimizer. Its BP rate peaks at 7.5×10^{-5} after 525 warmup updates and decays over the final 1,050 updates. That peak was selected from three candidates using training loss. These recipe and budget differences prevent attributing the ordering in the table to recurrence alone.

Table 7: **Single-switch and recurring schedules at two stages.** Each row is one run after 17,500 updates; purely local learning reaches 49.65%. Local-to-BP continuations start a fresh optimizer.

Allocation	BP updates	BP (%)	Accuracy (%)
BP opening, then local	2,000	11.43	51.79
Local opening, then BP	15,500	88.57	52.41
Local 7,000, then BP	10,500	60.00	50.18
Sparse Tidal	1,170	6.69	51.75
Tidal, 1,000 local / 100 BP	2,500	14.29	52.11
Full BP throughout	17,500	100.00	51.88

[2]The final evaluation log for this continuation reports task metrics to four decimal places, bounding rounding error in the mean by 0.005 percentage points.

B.8 UNEVEN PARTITIONS

The comparisons in Table 1 match model, data source, token batch and optimizer between local learning and Tidal within each partition. All four runs use peak rate 6×10^{-4}, floor 6×10^{-5}, 1,000 warmup updates and 16,500 cosine-decay updates. The $[8, 3, 3, 2]$ partition uses the sparse 1,170-BP schedule; $[2, 14]$ uses the dense 9,500-BP schedule. Their different rate from the equal-partition Tidal runs precludes treating the cross-partition comparison as an isolated effect of partition shape.

C SHORTER-BP COMPARISONS IN VISION MODELS

These studies test whether smaller settings reproduce both recovery over local learning and an advantage over BP trained for the same BP budget.

C.1 FINAL ACCURACY AND TRAINING TRAJECTORIES

We compare local learning throughout, full BP throughout, recurring Tidal, and shorter BP in three settings, each with three training seeds. The full training horizon is 14,040 updates for pretrained ViT-Base on CIFAR-100 and CINIC-10, and 35,000 updates for ResNet-101 trained from scratch on CIFAR-100. Tidal and shorter BP use 4,212 BP updates in each ViT setting and 10,500 in ResNet. The ViT rate is constant after warmup, so the BP prefix is also the shorter recipe. For ResNet, we evaluate both the inherited prefix and a separate short run that completes cosine decay at update 10,500.

Table 8: **Short BP has higher mean accuracy than Tidal in these smaller settings.** Final test accuracy averaged over three runs, with float32 inference and argmax prediction. CINIC uses the 20,000 CIFAR-origin test images. The last column is the 95% interval for Tidal minus short BP.

Setting	BP budget	Local	Tidal	Short BP	Full BP	Difference interval
CIFAR-100, ViT	4,212	83.82	91.01	92.24	92.62	$[-1.99, -0.47]$
CINIC-10, ViT	4,212	94.93	98.26	98.55	98.75	$[-0.63, 0.05]$
CIFAR-100, ResNet-101	10,500	71.18	75.28	77.79	79.51	$[-3.44, -1.58]$

Short BP has higher accuracy in every seed (Tables 8 and 9). Its mean advantage is 1.23 points for CIFAR ViT, 0.29 for CINIC ViT and 2.51 for ResNet; the CINIC difference remains uncertain across three seeds. For ResNet, completing decay changes the comparison: the inherited BP prefix reaches 55.74%, the short recipe 77.79%, and Tidal 75.28%. Exceeding a prefix of a longer BP run does not establish an advantage over BP optimized for the shorter horizon.

Accuracy decreases across all 66 observed local intervals in the two ViT studies and all 45 in the ResNet study (Figure 6). These intervals are repeated observations within three runs per setting. In these recipes, recovering a local-learning deficit does not yield an advantage over shorter BP, unlike the sparse 1B result.

Table 9: Individual final test accuracies for Table 8, in percent.

Setting	Seed	Local	Tidal	Short BP	Full BP
CIFAR-100, ViT	45	83.51	90.91	92.36	92.62
CIFAR-100, ViT	46	84.14	90.99	92.35	92.73
CIFAR-100, ViT	47	83.80	91.14	92.02	92.51
CINIC-10, ViT	45	94.61	98.39	98.53	98.81
CINIC-10, ViT	46	95.35	98.24	98.61	98.72
CINIC-10, ViT	47	94.81	98.14	98.51	98.72
CIFAR-100, ResNet-101	50	70.90	75.24	78.02	79.61
CIFAR-100, ResNet-101	51	71.25	74.88	77.54	79.79
CIFAR-100, ResNet-101	52	71.38	75.73	77.81	79.14

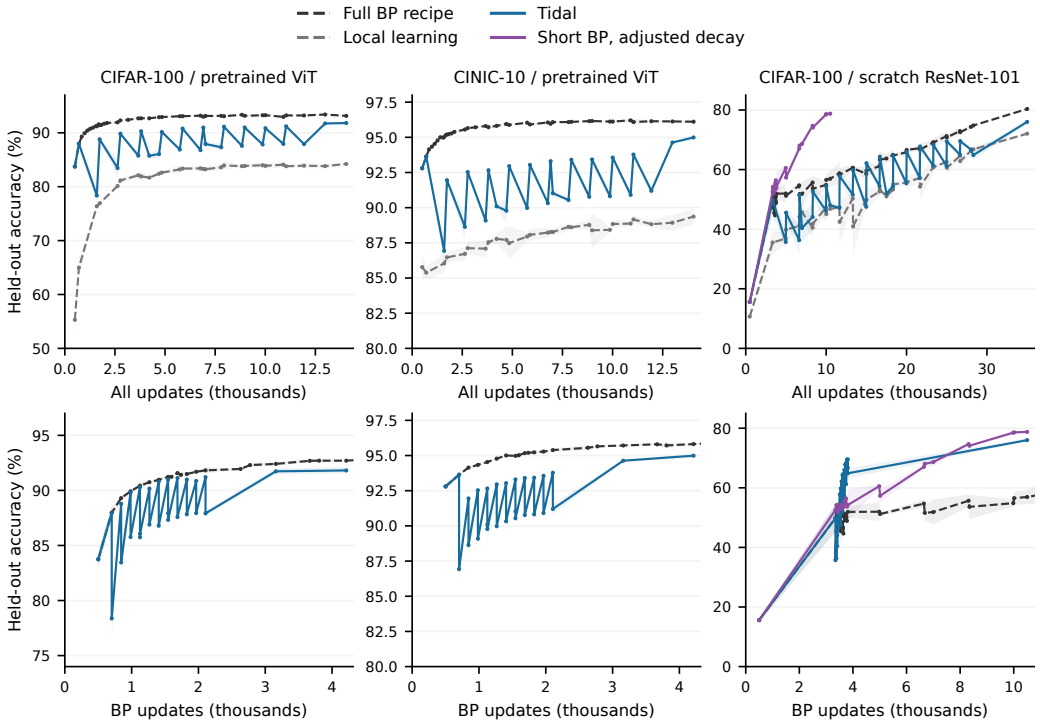


Figure 6: **Training trajectories in the smaller settings.** Top: accuracy by total updates. Bottom: the same observations by cumulative BP updates, restricted to Tidal's BP budget. Vertical Tidal segments contain local updates only. Means and sample standard deviations over three seeds use held-out training data: 5,000 CIFAR images for ViT, 90,000 CINIC validation images and a 4,500-image CIFAR subset for ResNet. ViT uses native BF16 evaluation with the first-of-top-five prediction rule; ResNet uses float32 argmax. These observations are separate from the final test scores in Table 8. The purple ResNet curve completes cosine decay at the shorter horizon.

C.2 Training and evaluation protocols

The ViT studies fine-tune ImageNet-21k pretrained ViT-Base, dividing its twelve transformer blocks into twelve stages (Dosovitskiy et al., 2021). Each nonfinal auxiliary predictor contains a newly initialized transformer block, normalization and a linear classifier applied to the class token. The main class token and positional embeddings remain fixed in every recipe, including full BP. SGD uses rate 0.01, momentum 0.9, no weight decay and batch size 128, with 500 warmup updates followed by a constant rate. CIFAR-100 uses 45,000 training and 5,000 validation images; CINIC-10 uses its 90,000-image training and validation splits (Darlow et al., 2018). Training uses random resized crops and horizontal flips at 224×224; evaluation uses deterministic resizing. All ViT comparisons use seeds 45–47. The recurring schedule has 702 opening BP updates, ten pulses totaling 1,404 BP updates, and 2,106 closing BP updates. The 9,828 local updates are divided across the eleven intervening phases; integer remainders are assigned to the earliest phases.

The ResNet comparison trains from scratch with 24 stages, using seeds 50–52 and the same 45,000-image CIFAR training subset (He et al., 2016). Its stem, 33 residual blocks and final classifier form 35 units: the first thirteen stages receive one unit each and the remaining eleven receive two. Each nonfinal auxiliary predictor adds a residual bottleneck block, global average pooling and a linear classifier. SGD uses peak rate 0.2, momentum 0.9, weight decay 5×10^{-4} and batch size 256. Training uses crops with four-pixel padding and horizontal flips; 500 warmup updates precede cosine decay through update 35,000. Tidal allocates 3,347 opening BP updates, fourteen pulses of 33 BP updates, and 6,691 closing BP updates. The 24,500 local updates are divided across fifteen phases, assigning integer remainders to the earliest phases. Short BP preserves the peak rate and warmup, completing decay at update 10,500.

Optimizer state and the learning-rate clock continue through switches. Auxiliary parameters update locally and are retained during BP. Gradient clipping uses norm 1.0 per stage in local mode and across the backbone in BP mode; auxiliary normalization buffers can also change during forward execution.

Final scores use float32 argmax on the official 10,000-image CIFAR test set or the 20,000 CIFAR-origin CINIC test images. The latter avoids CINIC's ImageNet-origin component, which may overlap ViT pretraining data. Training curves use separate validation data and the native conventions specified in Figure 6. Confidence intervals are two-sided Student-t intervals over the three within-seed accuracy differences.

D Data and analysis

The accompanying `data/tidal_sources.json` manifest identifies source revisions, evaluations, configurations and schedules. `scripts/tidal_evidence.py` reproduces task means, BP counts and schedule comparisons; `scripts/tidal_training_curves.py` reproduces the trajectories and shorter-BP comparisons, checking exact prediction counts where available. Their evidence files retain the source records and hashes, distinguish training observations from final test evaluations, and mark unavailable fields explicitly.