DYNAMIC SPARSE TRAINING FOR SUSTAINED PLASTICITY UNDER NON-STATIONARY DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep neural networks trained on non-stationary data, while retaining previously observed samples, represent a continual learning (CL) setting in which loss of plasticity remains an important failure mode, and catastrophic forgetting is not the primary challenge. Plasticity is the ability of a network to continue learning effectively from new data; its loss shows up as warm-started networks generalizing worse than networks trained from scratch on the same data. Weight sparsity has been used in CL to reduce task interference and mitigate catastrophic forgetting, but its role in preserving plasticity remains less understood, particularly when previously observed data remains available. We study whether unstructured sparse training (static sparse training (SST), gradual magnitude pruning (GMP), and dynamic sparse training (DST)) can mitigate plasticity loss in data-incremental learning settings. Across sample-incremental and class-incremental continual visual learning on CIFAR-10, CIFAR-100, and Tiny ImageNet, DST, in particular Rigging the Lottery (RigL), at moderate sparsities outperforms dense baselines, while in continual language fine tuning on TRACE, RigL at moderate sparsity likewise improves over dense training, with lower sparsity providing no advantage. RigL also compares favorably with dense training at most fractions of retained past data. Treating DST as an alternative training procedure rather than another plasticity intervention, we combine it with other plasticity improving interventions; both perform better on top of RigL than on top of dense training in continual visual learning and continual language fine tuning. Overall, our results suggest that dynamically evolving a sparse topology can help sustain plasticity across architectures and learning domains while remaining compatible with existing interventions for plasticity loss.

1 INTRODUCTION

Models deployed in real-world settings are periodically retrained as new, task-relevant data is acquired (Masana et al., 2022; Wang et al., 2024): visual recognition systems encounter new images and categories (Ash & Adams, 2020; Qazi et al., 2026), and language models are adapted to additional fint-tuining tasks (Prabhu et al., 2023; Wang et al., 2023). Continual learning (CL) frames this problem through the stability–plasticity tradeoff (Mermillod et al., 2013; Carpenter & Grossberg, 1987); CL conventional protocols assume that most past data is unavailable, making catastrophic forgetting the dominant failure mode (Kirkpatrick et al., 2017; Rebuffi et al., 2017; Rolnick et al., 2019). In many deployments this assumption is unnecessarily restrictive: storage is cheap relative to compute, and the goal is to maximize performance rather than to minimize memory (Prabhu et al., 2023; Cho et al., 2026). Since replaying even a fraction of past data largely mitigates forgetting (Rebuffi et al., 2017; Zhou et al., 2024), the central question shifts from what a model retains to how effectively it continues to learn.

We call this regime, in which new data arrive at each stage and past data is retained, the *data-incremental* setting; its natural policy is to *warm-start* each stage from the preceding one's parameters (Ash & Adams, 2020; Prabhu et al., 2023; Cho et al., 2026). Yet the resulting networks generalize worse than networks trained from scratch on the same accumulated data, even at comparable training loss (Ash & Adams, 2020; Berariu et al., 2021; Shin et al., 2024); since the data is identical, the gap must originate in the inherited parameters (Cho et al., 2026; Han et al., 2026). This degradation is described as a loss of *plasticity*, the capacity to learn effectively from later data

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.

(Lyle et al., 2023; Dohare et al., 2024). Under plasticity loss scenarios, *generalizability* rather than *trainability* is important as the training loss on new data decreases, but held-out performance does not improve accordingly (Lee et al., 2024; Lyle et al., 2025). We therefore use held-out performance as our primary measure and report plasticity metrics as proxy feedback (Lyle et al., 2023). Existing remedies shrink, perturb, transform, or reset the learned weights (Ash & Adams, 2020; Shin et al., 2024; Han et al., 2026; Lee et al., 2024; Cho et al., 2026), or reinitialize dormant or low-utility units (Sokar et al., 2023; Dohare et al., 2024), but leave the topology, the set of active connections, fixed. sparse training (ST) is a candidate set of methods which alter the topology of the network. In CL, however, sparsity has mainly been used to isolate task-specific parameters and protect prior knowledge (Gurbuz & Dovrolis, 2022; Wang et al., 2022; Yildirim et al., 2024). Sparse training has shown mixed but promising results against plasticity loss in deep reinforcement learning (Sokar et al., 2022; Obando-Ceron et al., 2024; Todorov et al., 2025; Graesser et al., 2022; Ganaie, 2026). Dynamic sparse training (DST) is a dynamic topology adaptation training method: it keeps the number of active weights fixed while periodically replacing low-magnitude connections with new ones, chosen at random (Sparse Evolutionary Training (SET)) or by gradient magnitude (Rigging the Lottery (RigL)), and on a fixed dataset can match or exceed the accuracy of dense training (Mocanu et al., 2018; Evci et al., 2020; Liu et al., 2021b). Dohare et al. (2024) suggest DST as a way to maintain plasticity, but neither line of work evaluates it in the data-incremental setting, where targets are fixed and only the dataset grows. Two properties make DST plausible there: its mask is learned jointly with the weights (Liu et al., 2021b), so it can be revised as new data arrive, and gradient-based regrowth, as in RigL, places new connections where the loss gradient on the current data is largest.

We evaluate DST in two domains. In vision, following the warm-starting setup of Lee et al. (2024), we train ResNet-18 on CIFAR-10, ViT-Tiny on CIFAR-100, and VGG-16 on Tiny ImageNet, comparing dense training with static sparse training (SST), gradual magnitude pruning (GMP), SET, and RigL (Zhu & Gupta, 2017; Mocanu et al., 2018; Evci et al., 2020). Each stage adds either new examples of a fixed set of classes (sample-incremental) or new classes while training continues on earlier ones (class-incremental). In continual fine-tuning, we fine-tune Qwen2.5-0.5B on the eight-task TRACE benchmark (Qwen Team, 2024; Wang et al., 2023), sparsifying only the gate-projection weights (Lu et al., 2026), and compare dense training with RigL, selected by a preliminary sweep over the sparse methods. Besides retaining all past data, we vary the retained fraction, which sets both storage and training cost. Finally, treating RigL as an alternative training procedure rather than another plasticity intervention, we apply other plasticity interventions on top of both dense training and RigL and evaluate performance.

Our contributions are as follows:

- **A controlled evaluation of sparse training for plasticity in the data-incremental setting.** To our knowledge, this is the first controlled comparison of sparse training methods in this setting; in both domains, RigL achieves the strongest overall performance among the methods we compare, including dense training (Sections 5.1–5.2).
- **RigL compares favorably with dense training at most data-retention levels.** RigL outperforms dense training at every retained fraction we evaluate in vision, and in language when greater than half of past data is retained (Section 5.3).
- **RigL appears complementary to existing plasticity interventions.** Both Frobenius–Isometry Reinitialization (FIRE) and full reinitialization perform better on top of RigL compared to dense training, and RigL with FIRE also outperforms RigL alone. This suggests that the benefits of RigL and of existing interventions can be combined (Section 5.4).

2 Related work

Warm-start studies establish that previous training can impair generalization on an expanded dataset, including under stationary sampling, and DASH and Hare & Tortoise respectively modify inherited weights and separate rapid adaptation from slower knowledge accumulation (Ash & Adams, 2020; Berariu et al., 2021; Shin et al., 2024; Lee et al., 2024). Computationally budgeted continual learning, Forget Forgetting, and FIRE study continued learning with substantial historical-data access, the latter two by introducing interventions on model weights (Prabhu et al., 2023; Cho et al., 2026; Han et al., 2026). Unlike these weight interventions, we change the training procedure itself, and evaluate

it both alone and combined with FIRE. SET and RigL establish sparse training with periodic pruning and regrowth, and in-time over-parameterization and gradient-flow analyses explain why evolving connectivity can behave differently from a fixed sparse parameterization (Mocanu et al., 2018; Evci et al., 2020; Liu et al., 2021c; Evci et al., 2022). GraNet's notion of pruning plasticity concerns recovery from parameter removal, rather than acquisition of subsequently arriving data (Liu et al., 2021a). Performance-oriented sparse training has also been studied through image-corruption robustness and deep reinforcement learning (Wu et al., 2025; Sokar et al., 2022). In the latter, broad comparisons and value-based studies show that pruning can be a strong alternative to both dense training and DST (Graesser et al., 2022; Obando-Ceron et al., 2024), while multi-task experiments connect sparsification to plasticity (Todorov et al., 2025). In value-based RL, DST improves plasticity proxies without attaining the highest returns, which GMP reaches Ganaie (2026); we ask how these methods compare under cumulative supervised learning, and use plasticity proxies to interpret performance rather than as the outcome.

In continual learning, PackNet protects task-specific subnetworks, DST has been studied within task-incremental parameter isolation, and NISPA and SparCL combine sparse rewiring with replay or rehearsal (Mallya & Lazebnik, 2018; Yildirim et al., 2024; Gurbuz & Dovrolis, 2022; Wang et al., 2022). Our comparison instead uses a shared model trained on the cumulative objective, without task-specific parameter protection. Continual adaptation of sparse language models has precedents in COPAL, DCFT, and GCL, while SpIEL evolves sparse parameter deltas over a pretrained model and SEFT adapts the topology of pruned LLMs (Malla et al., 2024; Wang et al., 2025; Wang & Yin, 2026; Ansell et al., 2024; Xiao et al., 2025). Our language experiment instead asks whether gate-projection RigL improves on dense fine-tuning as the eight TRACE instruction datasets accumulate under full historical-task-data retention (Wang et al., 2023; Cho et al., 2026). Feature renewal, regularization toward initialization, plasticity injection, and partial resets provide several ways to modify a model's capacity for subsequent learning (Dohare et al., 2024; Sokar et al., 2023; Kumar et al., 2025; Nikishin et al., 2023; 2022), and retraining on stored examples has precedents in GDumb (Prabhu et al., 2020). Lottery tickets show that learned sparse connectivity can be retrained from its original initialization (Frankle & Carbin, 2019). One Loop, Two Gains is particularly close: it integrates pruning and sparse retraining into active acquisition on growing labeled pools and includes a dense–RigL comparison whose ordering changes with the acquisition budget (Tscheschner et al., 2026). Our increments are prescribed, and we compare continued training with FIRE and full resets; in the reset condition, RigL inherits the preceding mask and continues rewiring during subsequent training. Our diagnostic analysis similarly draws on work separating trainability, generalization, and task performance (Lyle et al., 2023; Lee et al., 2024; Lyle et al., 2025; Ganaie, 2026).

The contribution of these comparisons is a connected empirical account of sparse training on growing supervised datasets: which procedures perform well, whether the dense–RigL finding extends across domains, and how it depends on historical retention and interventions on inherited weights.

3 METHODS

We compare dense training with three approaches to unstructured sparse training that differ in when the sparse connectivity is chosen and whether it changes during training. SST fixes the connectivity at initialization, GMP removes connections progressively and never restores them, and DST keeps the number of active connections fixed while periodically replacing some of them.

3.1 UNSTRUCTURED SPARSITY AND PRUNING CRITERIA

Let W^l be the weight tensor of a sparsified layer l, with n^l entries. A binary mask $M^l \in \{0,1\}^{n^l}$ determines which weights are active, and the forward pass is computed with the effective weights $\widetilde{W}^l = M^l \odot W^l$; all other parameters remain dense. The sparsity of layer l is $s^l = 1 - \|M^l\|_0/n^l$, and the overall sparsity s is the fraction of masked weights across all sparsified layers. The mask acts on individual weights, so the sparsity is unstructured, giving fine-grained control over the network topology. Pruning chooses a mask by giving each weight a saliency score that estimates the increase in loss caused by removing it, and keeping the $(1-s^l)\,n^l$ highest-scoring weights in each layer. The simplest criterion is magnitude, $|w_i|$, which assumes that small weights contribute little to a layer's output (Han et al., 2015).

Layer-wise allocation. The target sparsity s can be distributed across layers uniformly, with $s^l = s$ for every layer, or with the Erdős–Rényi-Kernel (ERK) allocation (Mocanu et al., 2018; Evci et al., 2020), which allocates higher density to layers with fewer parameters. For a convolutional layer with c_{in} input channels, c_{out} output channels, and a $\kappa_h \times \kappa_w$ kernel,

$$1 - s^l = \min\left(1,\ \epsilon \frac{c_{\text{in}} + c_{\text{out}} + \kappa_h + \kappa_w}{c_{\text{in}}\, c_{\text{out}}\, \kappa_h\, \kappa_w}\right), \tag{1}$$

where the kernel terms are omitted for linear layers and ϵ is chosen so that the overall sparsity equals s. In our experiments, RigL uses ERK and the other sparse methods use the uniform allocation.

3.2 Static sparse training

SST prunes once, at initialization, using magnitude as the saliency criterion, and keeps the resulting mask fixed for the whole of training. Since W_0^l is random, the mask chosen at initialization does not account for the input data and mostly leads to suboptimal performance.

3.3 Gradual magnitude pruning

GMP (Zhu & Gupta, 2017) trains the network densely until step t_0. From then on, every Δt steps, each layer removes its smallest-magnitude active weights until it reaches the target sparsity

$$s_t = s_f \left[1 - \left(1 - \frac{t - t_0}{t_f - t_0}\right)^3\right], \qquad t_0 \le t \le t_f, \tag{2}$$

which is held at its final value s_f after t_f. The cubic schedule removes many weights early, when redundant weights are plentiful, and fewer as sparsity approaches s_f, leaving time for the remaining weights to compensate. Removed weights are never restored. The hyperparameters are s_f, the pruning window $[t_0, t_f]$, and the pruning interval Δt.

3.4 Dynamic sparse training

DST methods keep the sparsity fixed during training but learn the sparse topology jointly with the weights. Training starts from a sparse mask, and every Δ steps each layer drops a fraction ζ_t of its active weights and grows the same number of inactive connections, so the number of active weights never changes. Between these updates, only the active weights are trained. Repeated drop and grow steps let the network explore many sparse topologies within a fixed parameter budget (Liu et al., 2021b).

SET (Mocanu et al., 2018) and RigL (Evci et al., 2020) both drop the active weights with the smallest magnitude, and differ in the criterion used to grow connections. SET grows inactive connections chosen uniformly at random, exploring new connectivity without reference to the loss. RigL grows the inactive connections with the largest gradient magnitude $|\partial \ell_t / \partial \widetilde{W}_i^l|$ on the current minibatch. These are the connections along which a gradient step would most reduce the loss. The dense gradient this requires is computed only at mask updates. Standard implementations initialize a regrown weight at zero (Evci et al., 2020). We instead restore the value the weight held when it was last dropped, or its initial value if it has never been active, so a connection dropped in an earlier stage re-enters with its learned value. Because dropped weights have small magnitude, regrowth perturbs the network's output only slightly.

The drop fraction decays within each stage k by cosine annealing,

$$\zeta_t = \frac{\zeta_0}{2}\left(1 + \cos\frac{\pi t}{\rho\, T_k}\right), \tag{3}$$

where t counts steps from the start of the stage and T_k is the number of steps in the stage. The topology therefore changes most at the start of each stage, when new data arrive, and is fixed for the final fraction $1 - \rho$ of the stage. The update interval Δ is set from a cadence parameter U, which we refer to as the total number of mask updates over training. The hyperparameters are the sparsity s, the initial drop fraction ζ_0, the cadence U, and ρ.

4 EXPERIMENTAL SETUP

4.1 DATA INCREMENTAL VISION

We evaluate sparse training methods over incremental data settings on vision and language benchmarks. Following is a description of the sample and class incremental settings on CIFAR-10, CIFAR-100 and Tiny ImageNet datasets borrowed from Han et al. (2026) we consider:

1. **Sample Incremental** : The training dataset is divided into chunks containing 10% of the total samples which are introduced in 10 stages to the training pipeline. The size of the training dataset gradually increases from 10% to 100% as data is added incrementally across the stages. The evaluation data is held constant at every training stage.
2. **Class Incremental**: In this setup, the training dataset is divided such that each chunk contains samples from only a fixed set of classes and is appended to the training dataset in stages as in the sample incremental case. We test this setup on CIFAR-100 and Tiny ImageNet datasets, where every chunk contains 5 and 10 classes respectively, leading to a 20-stage training pipeline. The evaluation dataset at a particular stage contains samples from all classes seen during training until that stage.

We train ResNet-18 (He et al., 2016), ViT-Tiny (Dosovitskiy et al., 2021a) and VGG-16 (Simonyan & Zisserman, 2015a) networks on CIFAR-10, CIFAR-100 and Tiny ImageNet datasets. The ResNet-18/CIFAR-10 benchmark is evaluated on the sample incremental setting while the other two benchmarks are evaluated on both sample and class incremental settings, consistent with the experiments in Han et al. (2026).

4.2 INCREMENTAL CONTINUAL LANGUAGE FINE TUNING

Langugae Benchmark: We evaluate on TRACE (Wang et al., 2023), a continual-learning benchmark of eight instruction-tuning tasks spanning classification, multi-hop reasoning, code completion, and summarisation. We use the released 5,000-example-per-task split and TRACE's default task order: C-STANCE, FOMC, MeetingBank, Py150, ScienceQA, NumGLUE-cm, NumGLUE-ds, 20Minuten. Most experiments use a three-task subset in sequence order FOMC $\rightarrow$ ScienceQA $\rightarrow$ NumGLUE-cm, chosen because all three are accuracy-scored and score exactly 0.000 zero-shot on every base model we study, so forward transfer is measured against a clean zero baseline rather than a confounded one; the full eight-task sequence is used where noted.

Models and training. We fully fine-tune Qwen2.5 base models at 0.5B and 3B parameters — no adapters or parameter-efficient methods. Training follows TRACE's own optimisation: AdamW, learning rate 1e-5 held constant with no warmup, zero weight decay, and an effective batch of 128 obtained by gradient accumulation. Prompts are truncated to 1,024 tokens and answers to 512, with length-grouped batching.

Sparse training. Sparsity is applied only to the gate projection matrix of every decoder block, leaving embeddings, the tied output head, attention projections, and normalisation layers dense

Evaluation and metrics. After each task boundary we evaluate every task on up to 500 held-out examples using greedy decoding, yielding a score matrix R where R(t,i) is performance on task i after training through task t.

5 RESULTS

We evaluate whether changing sparse connectivity improves predictive performance as training data accumulates. We first compare training procedures in vision and language, then examine how the dense–RigL difference depends on historical-data retention and interventions on inherited weights. Finally, we use plasticity diagnostics to assess which performance differences are reflected in the measured activity, representations, and weight geometry.

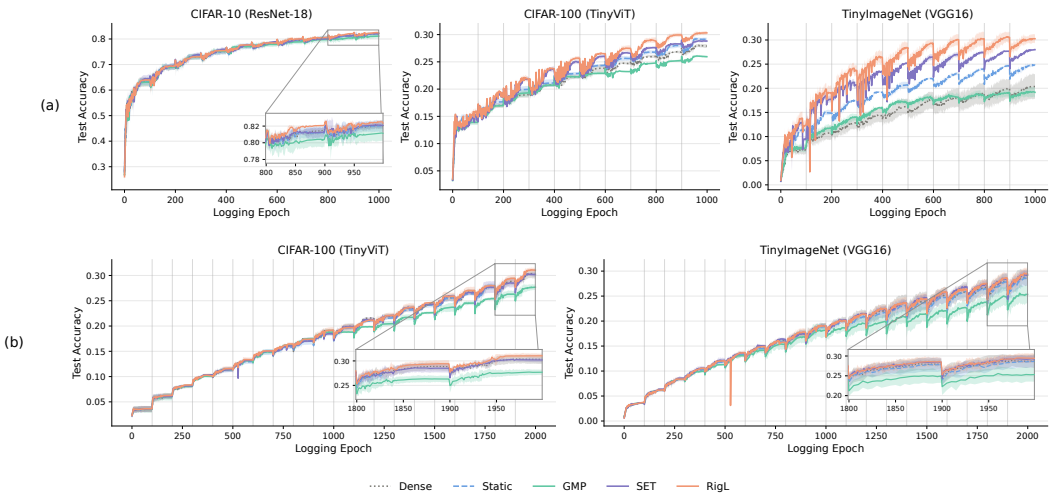


Figure 1: **Comparison of different sparse training methods** a) Sample Incremental setting. b) Class Incremental setting. Bands denote one standard deviation. Best DST configuration surpasses dense over all the benchmarks

5.1 Comparison of Sparse Training Methods on Data Incremental Vision Benchmarks

We compare dense training, SST, GMP, SET, and RigL over the sample and class incremental vision benchmarks stated in section 4. For DST methods and GMP, we sweep across different sparsity and mask update values as specified in table Table 3 and report the configurations with the best performance. In general, the performance of DST methods does not vary much with the initial drop fraction; yet we sweep across a low and a high value of initial drop fraction across both SET and RigL. For SST, the only hyperparameter swept is the sparsity. The reported accuracy values are average values across 3 random seeds and the shaded region around the curves shows the standard deviation around the mean.

RigL achieves the highest mean final accuracy across all the evaluated dataset–architecture pairs. SET matches dense in some settings while surpassing over it in the others. The gap between dense and RigL varies across the different benchmarks and settings. For example, in case of sample incremental setting, the accuracy at convergence is slightly better than dense for the CIFAR-10/ResNet-18 benchmark while RigL has a mean accuracy 9 percentage points higher than dense on the Tiny ImageNet/ VGG16 benchmark. But for the class incremental setup, the performance gap between RigL remains very small across both the benchmarks.

Table 5 (Appendix A) details the configurations for DST methods that achieve the best overall performance. The sparsity at which DST methods surpass dense is different for different benchmarks and settings. The optimal sparsity in case of ResNet-18 is 10%, TinyVIT is 50% and VGG16 is 90% for the sample incremental setting, hinting at a large parameter redundancy in the VGG networks. These findings highlight the sensitivity of DST methods to the benchmark and downstream task.

Another surprising observation is that SST matches dense training or even surpasses it in some cases. On the other hand, GMP consistently under-performs compared to dense in all settings. These observations suggest that on vision benchmarks with expanding training data, sparse training from scratch can enhance generalization, more so if the sparse topology is adapted using DST. But inducing sparsity gradually following a small phase of dense training, as in the case of GMP, makes generalization worse compared to dense.

5.2 Extending the comparison to continual language fine-tuning tuning

We next examine whether this advantage transfers from vision to the sequential adaptation of a pretrained language model, where fine-tuning tasks arrive one after another. On the full eight-task

Table 1: **Continual instruction tuning on the full TRACE sequence.** Qwen2.5-0.5B with full cumulative retention. Sparsity applies only to eligible gate-projection weights. OP is the mean of task-specific scores on a 0–1 scale, not a common accuracy metric.

Method	C-STANCE	FOMC	MeetingBank	Py150	ScienceQA	NumG-cm	NumG-ds	20Minuten	OP
Dense	.502	.633	.478	**.588**	.876	.420	.566	**.395**	.5572
RigL, $s = .1$	.486	.613	**.484**	.579	.872	.432	**.597**	.391	.5567
RigL, $s = .3$	**.528**	**.641**	**.484**	.580	**.892**	**.481**	.557	**.395**	**.5699**

Table 2: **Effect of data retention on TRACE.** Overall performance (OP) of dense training and RigL on a three-task subset of the TRACE benchmark for data retention fractions $r \in \{0.2, 0.5, 0.8\}$. Higher is better; the better method at each r is shown in bold.

	Data retention fraction r		
Method	0.2	0.5	0.8
Dense	**.5714**	**.5968**	.6050
RigL	.5709	.5866	**.6320**

TRACE sequence with Qwen2.5-0.5B under full retention of previous-task data, gate-projection pruned RigL at sparsity $= 0.3$ improves overall performance (OP) over dense fine-tuning (Table 1). Because OP is the mean of heterogeneous per-task scores on a 0–1 scale rather than a single accuracy metric, we read this margin on the benchmark's aggregate scale rather than as an accuracy improvement, and the gain is not distributed uniformly across the sequence: relative to dense fine-tuning, RigL improves the majority of task scores while trailing on two of the eight (Table 1). The size of this gain depends on the sparsity level. At the lowest sparsity we consider, the sparse model falls just below the dense reference, both on the full sequence (Table 1) and on the three-task selection sequence (Table 7). On that subset, an intermediate sparsity of $s = 0.3$ attains the highest OP, narrowly ahead of the higher setting and above the dense baseline. The choice of sparse-training algorithm, rather than sparsity alone, determines whether the approach helps. On the same three-task subset, we compare RigL at sparsity $= 0.3$ against three other gate-projection sparse methods at matched density — GMP, SST, and SET alongside dense fine-tuning (Table 8). Only RigL and GMP exceed the dense baseline in OP, and RigL is the strongest arm of the sweep; SST and SET both fall below dense. The gain is thus specific to how connectivity is selected and updated during training and is not a generic consequence of sparsity. RigL, the method carried over from the vision experiments, remains the best choice under sequential fine-tuning.

This advantage is not confined to the smallest model. We repeat the three-task comparison at a larger scale, fine-tuning Qwen2.5-3B alongside the 0.5B model under the same protocol (Table 9). At both sizes, gate-projection RigL at sparsity $= 0.3$ improves OP over dense fine-tuning, and the margin does not diminish with scale: the larger model retains an OP advantage comparable to that seen at 0.5B. The persistence of the effect across an order-of-magnitude change in parameter count indicates that the benefit of dynamic sparse connectivity is not an artifact of limited capacity in the smaller model, and that RigL remains the preferred arm as model size increases.

5.3 RigL is the cost-efficient choice at every retention fraction

We next consider the dense training and RigL comparison jointly in terms of performance and cost. In this incremental setting the dominant cost is the compute spent revisiting historical data: with epochs and batch size fixed, retaining a fraction r of earlier examples sets the number of optimizer updates and hence the GPU cost of each training round. This framing raises two coupled questions. First, at what retention level is performance best i.e. how much historic data do we need to keep on seeing an improvement in performance. Second, if cost pressures force r to be small, which procedure makes better use of a restricted budget? We therefore vary r, the fraction of earlier examples kept for training, holding epochs and batch size fixed so larger retained sets also yield more optimizer updates and report final accuracy on Tiny ImageNet/VGG-16 in both sample- and class-incremental settings (Figure 2), alongside overall performance on a three task subset of the TRACE language benchmark (Table 2).

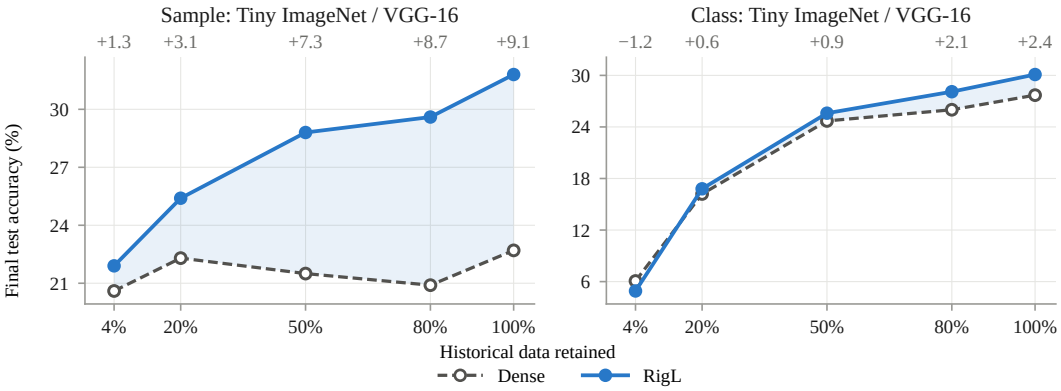


Figure 2: **Dependence on historical-data retention.** Final test accuracy on Tiny ImageNet/VGG-16 as the retained fraction r of earlier data varies; numbers give the RigL and dense difference. Epochs and batch size are fixed, so greater retention also means more optimizer updates.

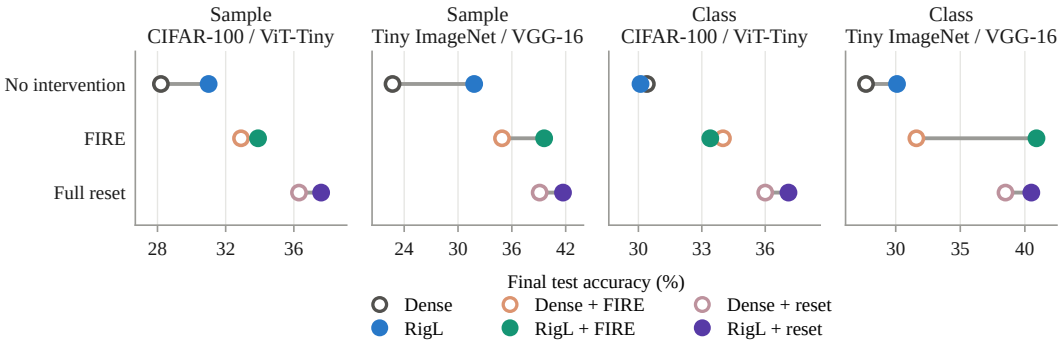


Figure 3: **Dense and RigL training with weight interventions.** Final test accuracy of dense training (hollow) and RigL (filled) without an intervention, with FIRE, and with a full reset at every stage boundary. Details in Appendix A.4.

Accuracy rises with the retained fraction for both procedures, so the best-performing regime is the one that retains the most data (Figure 2, Table 2). RigL gains more from each increase in r, so the gap widens with retention and is largest at full retention, in both image settings and on the TRACE subset. At the high-retention end, where performance is best, RigL is clearly stronger; at low retention the two are essentially level. RigL is therefore never the worse choice: it matches dense on a tight budget and pulls ahead on the budget that yields the best models.

5.4 Adding plasticity interventions on top of RigL

We next test whether the advantage of RigL remains when it is combined with interventions designed to restore plasticity in data-incremental learning. We consider two interventions applied at every stage boundary: FIRE (Han et al., 2026), which transforms the weight values, and a full reset, in which both procedures restore their saved initial parameters and non-mask buffers and renew optimizer state. We compare dense training and RigL with no intervention, with FIRE, and with full reset on sample-incremental CIFAR-100/ViT-Tiny (Figure 3). This design separates the benefit of replacing the training procedure from the benefit of adding an intervention to it.

FIRE improves performance for both dense and RigL training, and RigL with FIRE exceeds both RigL alone and dense with FIRE. The first comparison shows that the weight intervention remains useful when connectivity is already dynamic; the second shows that applying FIRE to the dense network does not remove the benefit of replacing dense training with RigL. These results support empirical complementarity: RigL changes the active connectivity during optimization, whereas FIRE transforms weight values while retaining the current mask in the sparse implementation. Full reset

yields the same qualitative picture. RigL with full reset substantially outperforms dense training with the same reset, improves on RigL without reset, and achieves the highest accuracy among the evaluated FIRE and reset configurations. The comparison with dense reset is particularly informative because that baseline retrains on the cumulative dataset without carrying forward previously optimized weights. The RigL advantage therefore cannot be explained solely as avoiding an unfavorable weight state inherited by the continuously trained dense model. In practice, adopting DST and applying a plasticity intervention are compatible choices: a stronger underlying training procedure remains beneficial after the intervention is introduced.

Plasticity diagnostics. To examine how well plasticity diagnostics account for this ordering, we measure them for dense, RigL, and their FIRE and full-reset combinations (Figure 12; definitions and additional measurements in Appendix B). Focusing on these conditions ties the measurements to the intervention results rather than requiring a single diagnostic to rank every sparse method. During continued training, RigL maintains fewer dormant neurons than the dense baseline as data accumulate. Under the activation-based dormancy criterion, this means that a smaller fraction of units has low activity relative to its layer, consistent with maintaining a larger fraction of responsive features. This links improved predictive performance to reduced inactivity, although activity alone does not measure how quickly those features can learn from the next increment. The remaining diagnostics characterize other aspects of the model state: feature rank describes the diversity of the measured representations, deviation from isometry characterizes weight geometry, and curvature and weight norm characterize the optimization landscape and parameter scale. Their changes provide additional context but vary with the intervention and do not yield a uniform ordering of methods. SFE measures weight change relative to a reference and is interpreted as a stability-related quantity rather than a direct measure of plasticity. We therefore assess these measurements jointly with performance.

6 Conclusion

We studied dynamic sparse training as an optimization procedure for learning from cumulatively growing datasets. Across the evaluated sample-incremental and class-incremental vision settings, RigL achieves the strongest overall performance among dense, static sparse, gradually pruned, and other DST methods. Gate-projection pruning usingf RigL also improves final aggregate performance over dense fine-tuning on the full TRACE sequence at the selected sparsity, extending the comparison from vision models trained from scratch to pretrained language-model adaptation. These findings identify active connectivity as an important training choice even when historical examples remain available. The retention and intervention experiments characterize the scope of this advantage. The dense–RigL gap grows with historical retention under a fixed-epoch protocol, which jointly increases data availability and training exposure. RigL remains beneficial when FIRE is applied to both procedures, and its combination with full weight reset achieves the highest performance among the evaluated intervention configurations. The latter result shows that the advantage survives discarding learned weight values between stages while preserving and continuing to update sparse connectivity. Thus, the benefit extends to both continued training and training with repeated weight resets, without isolating mask inheritance as its cause. Plasticity diagnostics provide a partial interpretation: favorable trends accompany the continued-training and FIRE results, but broadly similar measurements under full reset coexist with a substantial performance difference. Explaining this difference requires measurements that more directly connect connectivity evolution to subsequent learning dynamics. The empirical conclusion is that RigL provides a strong alternative to dense training in the studied cumulative-data protocols, with benefits that remain compatible with interventions on the weights. Establishing the extent of these benefits across larger models, alternative task sequences, and matched optimization budgets is a natural next step.

REFERENCES

Alan Ansell, Ivan Vulić, Hannah Sterz, Anna Korhonen, and Edoardo M. Ponti. Scaling sparse fine-tuning to large language models. *arXiv preprint arXiv:2401.16405*, 2024. URL https://arxiv.org/html/2401.16405.

Jordan T. Ash and Ryan P. Adams. On warm-starting neural network training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Tudor Berariu, Wojciech Czarnecki, Soham De, Jörg Bornschein, Samuel Smith, Razvan Pascanu, and Claudia Clopath. A study on the plasticity of neural networks. *arXiv preprint arXiv:2106.00042*, 2021. URL https://arxiv.org/abs/2106.00042.

Gail A. Carpenter and Stephen Grossberg. Art 2: self-organization of stable category recognition codes for analog input patterns. *Appl. Opt.*, 26(23):4919–4930, Dec 1987. doi: 10.1364/AO.26.004919. URL https://opg.optica.org/ao/abstract.cfm?URI=ao-26-23-4919.

Dongkyu Cho, Taesup Moon, Rumi Chunara, Kyunghyun Cho, and Sungmin Cha. Forget forgetting: Continual learning in a world of abundant memory. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/17fc58b1ba92abbb72c5f457e140d851-Abstract-Conference.html.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632:768–774, 2024.

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*, 2021a. URL https://openreview.net/forum?id=YicbFdNTTy.

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 (ICLR)*, 2021b.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning (ICML)*, 2020.

Utku Evci, Yani Ioannou, Cem Keskin, and Yann Dauphin. Gradient flow in sparse neural networks and how lottery tickets win. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6577–6586, 2022. doi: 10.1609/aaai.v36i6.20611. URL https://ojs.aaai.org/index.php/AAAI/article/view/20611.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Muhammad Athar Ganaie. Dynamic sparse training breaks the correlation between plasticity and reward in deep reinforcement learning. Master of science thesis, University of Calgary, February 2026. URL https://ucalgary.scholaris.ca/items/5ab84f67-9d05-4dbd-b703-1658eadb61d5.

Laura Graesser, Utku Evci, Erich Elsen, and Pablo Samuel Castro. The state of sparse training in deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2022.

Mustafa B. Gurbuz and Constantine Dovrolis. NISPA: Neuro-inspired stability-plasticity adaptation for continual learning in sparse networks. In *International Conference on Machine Learning*, 2022. URL https://proceedings.mlr.press/v162/gurbuz22a.html.

Isaac Han, Sangyeon Park, Seungwon Oh, Donghu Kim, Hojoon Lee, and Kyung-Joong Kim. FIRE: Frobenius-isometry reinitialization for balancing the stability-plasticity tradeoff. In *International Conference on Learning Representations (ICLR)*, 2026.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Saurabh Kumar, Henrik Marklund, and Benjamin Van Roy. Maintaining plasticity in continual learning via regenerative regularization. In *Conference on Lifelong Learning Agents (CoLLAs)*, 2025.

Mike Lasby. Sparsimony: Dynamic sparse training and pruning algorithms for PyTorch. `https://github.com/mklasby/sparsimony`, 2024. Version 0.2.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. Technical report, Stanford University, CS231N Course Report, 2015.

Hojoon Lee, Hyeonseo Cho, Hyunseung Kim, Donghu Kim, Dugki Min, Jaegul Choo, and Clare Lyle. Slow and steady wins the race: Maintaining plasticity with hare and tortoise networks. In *International Conference on Machine Learning (ICML)*, 2024.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Zahra Atashgahi, Lu Yin, Huanyu Kou, Li Shen, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Sparse training via boosting pruning plasticity with neuroregeneration. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021a.

Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Do we actually need dense over-parameterization? In-Time over-parameterization in sparse training. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6989–7000. PMLR, 2021b.

Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In *International Conference on Machine Learning (ICML)*, 2021c.

Jack Lu, Zhenbang Yang, Mike Lasby, Tejas Pote, Yani Ioannou, and Mengye Ren. The surprising effectiveness of deleting weights in LLM reasoning and adaptation. In *ICML 2026 Workshop on Foundations of Deep Generative Models: Understanding Memorization, Generalization, and Reasoning*, 2026. URL `https://openreview.net/forum?id=JnfolztAV3`.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *International Conference on Machine Learning (ICML)*, 2023.

Clare Lyle, Zeyu Zheng, Khimya Khetarpal, Hado van Hasselt, Razvan Pascanu, James Martens, and Will Dabney. Disentangling the causes of plasticity loss in neural networks. In *Conference on Lifelong Learning Agents (CoLLAs)*, 2025.

Srikanth Malla, Joon Hee Choi, and Chiho Choi. COPAL: Continual pruning in large language generative models. In *International Conference on Machine Learning*, 2024. URL `https://proceedings.mlr.press/v235/malla24a.html`.

Arun Mallya and Svetlana Lazebnik. PackNet: Adding multiple tasks to a single network by iterative pruning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification, 2022. URL https://arxiv.org/abs/2010.15277.

Martial Mermillod, Aurélia Bugaiska, and Patrick BONIN. The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in Psychology*, Volume 4 - 2013, 2013. ISSN 1664-1078. doi: 10.3389/fpsyg.2013.00504. URL https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2013.00504.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9(1):2383, 2018.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2022.

Evgenii Nikishin, Junhyuk Oh, Georg Ostrovski, Clare Lyle, Razvan Pascanu, Will Dabney, and André Barreto. Deep reinforcement learning with plasticity injection. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Johan Obando-Ceron, Aaron Courville, and Pablo Samuel Castro. In value-based deep reinforcement learning, a pruned network is a good network. In *International Conference on Machine Learning (ICML)*, 2024.

Ameya Prabhu, Philip H. S. Torr, and Puneet K. Dokania. GDumb: A simple approach that questions our progress in continual learning. In *European Conference on Computer Vision*, 2020. URL https://www.ecva.net/papers/eccv_2020/papers_ECCV/papers/123470511.pdf.

Ameya Prabhu, Hasan Abed Al Kader Hammoud, Puneet K. Dokania, Philip H. S. Torr, Ser-Nam Lim, Bernard Ghanem, and Adel Bibi. Computationally budgeted continual learning: What does matter? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Prabhu_Computationally_Budgeted_Continual_Learning_What_Does_Matter_CVPR_2023_paper.html.

Mohammad Areeb Qazi, Anees Ur Rehman Hashmi, Santosh Sanjeev, Ibrahim Almakky, Numan Saeed, Camila González, and Mohammad Yaqub. Continual learning in medical imaging: A survey and practical analysis. *ACM Comput. Surv.*, 58(8), February 2026. ISSN 0360-0300. doi: 10.1145/3785663. URL https://doi.org/10.1145/3785663.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Baekrok Shin, Junsoo Oh, Hanseul Cho, and Chulhee Yun. DASH: Warm-starting neural network training in stationary settings without loss of plasticity. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015a. URL http://arxiv.org/abs/1409.1556.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015b.

Ghada Sokar, Elena Mocanu, Decebal Constantin Mocanu, Mykola Pechenizkiy, and Peter Stone. Dynamic sparse training for deep reinforcement learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2023.

Aleksandar Todorov, Juan Cardenas-Cartagena, Rafael F. Cunha, Marco Zullich, and Matthia Sabatelli. Sparsity-driven plasticity in multi-task reinforcement learning. *Transactions on Machine Learning Research*, 2025.

Benedikt Tscheschner, Eduardo Veas, and Marc Masana. One loop, two gains: Can active learning win the lottery for free? *arXiv preprint arXiv:2609.10311*, 2026. URL https://arxiv.org/html/2609.10311v1.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application, 2024. URL https://arxiv.org/abs/2302.00487.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, and Xuanjing Huang. TRACE: A comprehensive benchmark for continual learning in large language models. *arXiv preprint arXiv:2310.06762*, 2023. URL https://arxiv.org/abs/2310.06762.

Yanzhe Wang and Baoqun Yin. GCL: Group-shared continual learning fine-tuning for sparse LLMs. *Neurocomputing*, 2026. URL https://doi.org/10.1016/j.neucom.2026.132918.

Yanzhe Wang, Yizhen Wang, and Baoqun Yin. DCFT: Dependency-aware continual learning fine-tuning for sparse LLMs. *Neurocomputing*, 2025. URL https://doi.org/10.1016/j.neucom.2025.129897.

Zifeng Wang, Zheng Zhan, Yifan Gong, Geng Yuan, Wei Niu, Tong Jian, Bin Ren, Stratis Ioannidis, Yanzhi Wang, and Jennifer Dy. SparCL: Sparse continual learning on the edge. In *Advances in Neural Information Processing Systems*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/80133d0f6eccaace15508f91e3c5a93c-Abstract-Conference.html.

Boqian Wu, Qiao Xiao, Shunxin Wang, Nicola Strisciuglio, Mykola Pechenizkiy, Maurice van Keulen, Decebal Constantin Mocanu, and Elena Mocanu. Dynamic sparse training versus dense training: The unexpected winner in image corruption robustness. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/92abec9d3f278c648dfe99c8b8f35954-Paper-Conference.pdf.

Qiao Xiao, Alan Ansell, Boqian Wu, Lu Yin, Mykola Pechenizkiy, Shiwei Liu, and Decebal Constantin Mocanu. Leave it to the specialist: Repair sparse LLMs with sparse fine-tuning via sparsity evolution. *arXiv preprint arXiv:2505.24037*, 2025. URL https://arxiv.org/html/2505.24037.

Murat Onur Yildirim, Elif Ceren Gok, Ghada Sokar, Decebal Constantin Mocanu, and Joaquin Vanschoren. Continual learning with dynamic sparse training: Exploring algorithms for effective model updates. In *Conference on Parsimony and Learning*, volume 234 of *Proceedings of Machine Learning Research*, pp. 94–107, 2024. URL https://proceedings.mlr.press/v234/yildirim24a.html.

Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Class-incremental learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9851–9873, December 2024. ISSN 1939-3539. doi: 10.1109/tpami.2024.3429383. URL http://dx.doi.org/10.1109/TPAMI.2024.3429383.

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

A IMPLEMENTATION DETAILS AND EVIDENCE STATUS

This working draft distinguishes supplied numerical tables, reconstructed plot values, and missing results. The source images are retained unchanged. The reconstruction script exports plot geometry and digitized values to CSV, with the originating file and extraction method. Reconstructed values are not substitutes for raw seed-level measurements: visual precision, incomplete runs, source aggregation, and plotting transformations can limit interpretation.

A.1 EXPERIMENTAL PROTOCOL AT A GLANCE

Tables 3–4 collect the settings used in the main comparisons. Run-specific overrides and incomplete exports remain identified in the final review notes.

Table 3: Vision protocol and sparse-training configuration.

Component	Configuration
Dataset / architecture	CIFAR-10 / ResNet-18; CIFAR-100 / ViT-Tiny; Tiny ImageNet / VGG-16 (Krizhevsky, 2009; Le & Yang, 2015; He et al., 2016; Dosovitskiy et al., 2021b; Simonyan & Zisserman, 2015b)
Data increments	Ten sample increments for all pairs; twenty class increments for CIFAR-100 and Tiny ImageNet
Historical data	Full cumulative retention, except in the retention ablation
Optimization	Adam; learning rate 10^{-3}; batch size 256; 100 epochs per stage; gradient clipping 0.5
Schedule / augmentation	Linear warmup over the first 10% of each stage; no data augmentation
Stage boundary	Fresh optimizer; weights retained unless a reset is applied
Sparse implementation	Sparsimony (Lasby, 2024); convolutional and linear weights masked, excluding the final classifier; biases and normalization remain dense
Layer allocation	ERK for RigL; uniform for the other sparse methods
Sweep parameters	Sparsities $\{.1, .3, .5, .7, .9\}$; drop fractions $\{.3, .7\}$; cadence inputs $\{10, 100, 1000\}$, with coverage varying by setting
Selection / evaluation	Validation-selected configurations; separate test set; final test accuracy and stage-wise trajectories
Replication	Broad vision comparisons: three seeds, mean and standard deviation; supplied FIRE comparisons: seed 0 except the three-seed dense reference; reset comparisons: seeds 0, 5, and 8

Table 4: Language protocol for the supplied TRACE experiments.

Component	Configuration
Model / hardware	Qwen2.5-0.5B; one NVIDIA H100 80GB SXM; float32
Task order	C-STANCE, FOMC, MeetingBank, Py150, ScienceQA, NumGLUE-cm, NumGLUE-ds, 20Minuten
Epochs by stage	5, 3, 7, 5, 3, 5, 5, 7
Optimization	AdamW; constant learning rate 10^{-5}; betas $(.9, .95)$; weight decay 0; gradient clipping 1; no warmup
Batching / lengths	Microbatch 4, accumulation 32 (effective batch 128); length-grouped batches; prompt cap 1024 and answer cap 512 tokens
Sparse scope	Decoder-block gate-projection weights only; other projections, embeddings, tied output head, and normalization remain dense
Initialization / growth	Magnitude-pruned initialization; ERK allocation; stored previous values used for regrowth
Topology schedule	Per-stage cosine drop schedule starting at .3; updates through 80% of planned stage steps; intervals 9 (full sequence) and 7 (subset); cadence inputs 600 and 100
Numerical guard	Non-finite-gradient steps skip optimizer and sparsifier updates; abort threshold 5%

Selection, evaluation, and interpretation. Sample-incremental vision evaluation uses the full test set throughout training. Class-incremental evaluation distinguishes seen-class and all-class accuracy; these coincide at the final stage.

The language subset uses FOMC, ScienceQA, and NumGLUE-cm in canonical relative order. It supports sparsity selection and extended-training ablations with a shared accuracy metric. Task outcomes were inspected during planning, and selection and reporting use the same evaluation split; the full sequence includes the three selection tasks. The comparison is therefore exploratory rather than an independent validation of the selected configuration. For task score $R_{t,i}$ after stage t, initialization score b_i, and T tasks, we use

$$\mathrm{OP} = \frac{1}{T}\sum_{i=1}^{T} R_{T,i}, \tag{4}$$

$$\mathrm{FWT} = \frac{1}{T-1}\sum_{i=2}^{T}(R_{i-1,i} - b_i). \tag{5}$$

The dense reference b_i uses the original pretrained checkpoint; RigL uses its magnitude-pruned initialization. FWT consequently measures improvement relative to each method's own starting model, so differences in FWT need not reflect differences in absolute pre-task performance. The primary language outcome is final OP.

A.2 Vision and sparse schedules

The vision trainer uses ten sample-incremental stages or twenty class-incremental stages with full historical retention. It recreates the optimizer at every stage and excludes the output classifier from masking. The source protocol uses Adam, learning rate 10^{-3}, batch 256, gradient clipping .5, and 100 epochs per stage. Best-configuration selection is validation-based according to the author, but its exact aggregation rule and the complete selected-configuration manifest are not present in the supplied plots.

For stage step counts T_k and total $T = \sum_k T_k$, the cadence is $\Delta = \max(1, \lfloor \lfloor \rho T \rfloor / U \rfloor)$. The inspected vision per-stage option restarts the sparsifier counter and uses horizon T_k, keeping Δ fixed. Language instead uses $\max(1, \lfloor \rho T_k \rfloor)$ with $\rho = .8$. Consequently, U is a cadence parameter, not an exact realized update count. Language sparsifier steps occur after successful optimizer updates, not microbatches; skipped non-finite updates do not advance the sparsifier. Previous-value regrowth restores stored parameter values, which need not remain equal to pretrained values after prior training.

Table 5: **Selected RigL configurations.** Sparsity s, drop fraction, and cadence parameter U of the best RigL configuration in each sweep, with its final test accuracy (%, mean $\pm$ standard deviation over three seeds). Class-incremental accuracy is measured on all classes.

Setting	Dataset / architecture	s	Drop fraction	U	Accuracy (%)
Sample	CIFAR-10 / ResNet-18	0.1	0.1	1000	82.48 ± 0.18
Sample	CIFAR-100 / ViT-Tiny	0.5	0.7	1000	30.31 ± 0.33
Sample	Tiny ImageNet / VGG-16	0.9	0.7	100	30.29 ± 0.90
Class	CIFAR-100 / ViT-Tiny	0.3	0.7	10	31.06 ± 0.46
Class	Tiny ImageNet / VGG-16	0.1	0.7	100	29.49 ± 0.12

A.3 Language configuration and supplementary scores

The full language setup uses Qwen2.5-0.5B, full cumulative data, final checkpoints, float32, and one H100 80GB SXM. Training uses AdamW with betas $(.9, .95)$, zero weight decay, gradient clipping 1, constant learning rate 10^{-5}, and effective batch 128. The optimizer object is retained while its state is cleared between tasks. Sparse results use only gate projections; the other projections and embeddings remain dense.

Task metrics are accuracy for C-STANCE, FOMC, ScienceQA, and both NumGLUE tasks; ROUGE-L for MeetingBank; edit similarity for Py150; and SARI for 20Minuten. Edit similarity and SARI are divided by 100 before aggregation. Generation is greedy, with token caps 8 for

Table 6: **Reconstructed vision endpoints (%).** Values at the full plotted horizon, rounded to two decimals. A dash means no trace reaches that horizon or the method is missing; no endpoint is extrapolated. These preliminary values still require raw-run verification.

Setting / dataset	Dense	Static	GMP	SET	RigL
Sample / CIFAR-10	81.98	82.04	81.14	82.08	82.36
Sample / CIFAR-100	27.90	29.12	25.95	28.81	30.32
Sample / Tiny ImageNet	20.36	24.73	19.16	27.99	30.26
Class / CIFAR-100	30.33	–	–	30.19	31.04
Class / Tiny ImageNet	28.73	30.05	–	30.03	–

Table 7: **Sparsity selection on a three-task TRACE subset.** FOMC → ScienceQA → NumGLUE-cm, cumulative retention. OP is mean accuracy.

Method	FOMC	ScienceQA	NumGLUE-cm	OP	FWT
Dense	.629	**.856**	.407	.6308	**.1270**
RigL, $s = .1$	.633	.852	.370	.6185	.0780
RigL, $s = .3$	.657	.840	**.420**	**.6390**	.1150
RigL, $s = .5$	**.663**	.842	.407	.6376	.0510

C-STANCE/FOMC, 16 for NumGLUE, 128 for Py150, 384 for MeetingBank/20Minuten, and 512 for ScienceQA in the inspected evaluator. Zero-shot evaluation occurs after sparse initialization; next-task evaluation supplies the pre-exposure scores needed for FWT. All language results use a single seed, and on the three-task subset the same evaluation split is used for selection and reporting.

A.4 Sparse FIRE and full weight reset

For effective weights $M \odot W$, the sparse FIRE adaptation computes

$$W \leftarrow \mathcal{O}(M \odot W)/\sqrt{\max(d, 10^{-8})}, \qquad W_{\text{eff}} \leftarrow M \odot W, \tag{6}$$

where d is density and $\mathcal{O}$ is the finite-iteration Newton–Schulz transformation with the implementation's scaling. The mask is unchanged; remasking need not preserve exact isometry. Convolutional weights are transformed slice-wise. For ViT-Tiny the selected projections are query and key; for VGG the selected modules are linear and convolutional layers. The source default is ten iterations, subject to run overrides.

Full reset copies all saved initial parameters, including classifier, biases, and normalization parameters, and restores non-mask buffers. The initial model is saved after sparsifier preparation. Masks are retained and optimizer state is renewed. The procedure does not draw fresh random weights at each boundary. Its performance does not isolate a causal benefit of mask inheritance; a fresh-mask experiment is optional future analysis.

Table 8: **Sparse method selection on a three-task TRACE subset.** All sparse arms use the eligible gate-projection weights at the stated sparsity s; ΔOP is relative to the dense baseline. Rows are ordered by OP.

Method	FOMC	ScienceQA	NumGLUE-cm	OP	ΔOP
RigL, $s = .3$	.657	.840	**.420**	**.6390**	+.0082
GMP, $s = .3$	**.679**	.828	.395	.6342	+.0034
Dense	.629	**.856**	.407	.6308	—
Static, $s = .3$	.653	.830	.370	.6179	−.0129
SET, $s = .3$	.625	.850	.346	.6069	−.0239

Table 9: **Model-scale comparison on a three-task TRACE subset.** Dense fine-tuning versus gate-projection RigL ($s = .3$) at two Qwen2.5 sizes.

Model	Method	FOMC	ScienceQA	NumGLUE-cm	OP
0.5B	Dense	.629	**.856**	**.407**	.6308
	RigL, $s = .3$	**.657**	.840	.420	**.6390**
3B	Dense	.661	**.950**	.506	.7058
	RigL, $s = .3$	**.698**	.926	**.543**	**.7223**

B DIAGNOSTIC DEFINITIONS AND REMAINING MEASUREMENT DETAILS

All diagnostics are computed on a fixed probe set of training images, 1,024 for ViT-Tiny and 4,096 for VGG-16, drawn once with a fixed seed and shared by all runs. They are measured at epoch 90 of every stage, unless stated otherwise.

Activity. A unit is dormant when its mean absolute activation on the probe set is at most $\tau = 0.1$ of the mean over its layer, and dead when its mean activation is non-positive. Both are reported as percentages pooled over all activation layers. The two criteria measure different quantities, so the dead fraction can exceed the dormant fraction, particularly under GELU activations in ViT-Tiny.

Representations. Rank estimates use the penultimate features of the probe set. With singular values $\sigma_1 \geq \cdots \geq \sigma_d$, srank and approximate rank are the number of singular values, or squared singular values, needed to reach 99% of their sum. Effective rank and RankMe are the exponential of the entropy of the normalized singular values and of their squares. Feature rank counts singular values above $0.01\sqrt{n}$ for n probe images. Feature norm is the mean ℓ_2 norm of the penultimate features.

Weight geometry. Following FIRE (Han et al., 2026), $\mathrm{DfI}(W) = \|W^\top W - I\|_F^2$, computed after rescaling W to unit root-mean-square singular value and averaged over weight matrices. For convolutions it is also computed per kernel slice, and for ViT-Tiny separately on the query and key projections that FIRE transforms. SFE is $\|W - W_0\|_F^2 / \|W_0\|_F^2$, measured relative to the initialization W_0. Weight norm is the ℓ_2 norm of the active weights.

Optimization. Gradient norm is the ℓ_2 norm of the training-loss gradient on the probe set. Gradient cosine is the cosine between this gradient and the one at the previous measurement (epoch 80), averaged over weight matrices. $\lambda_{\max}$ is the largest eigenvalue of the training-loss Hessian on 256 probe images, estimated by 20 power iterations, and the Hessian trace is a Hutchinson estimate with 50 Rademacher vectors on the same images. Both are measured at the start of each stage. These quantities are complementary proxies, not a uniform ordering of plasticity.

C SUPPLEMENTARY CURVES AND INCOMPLETE SWEEPS

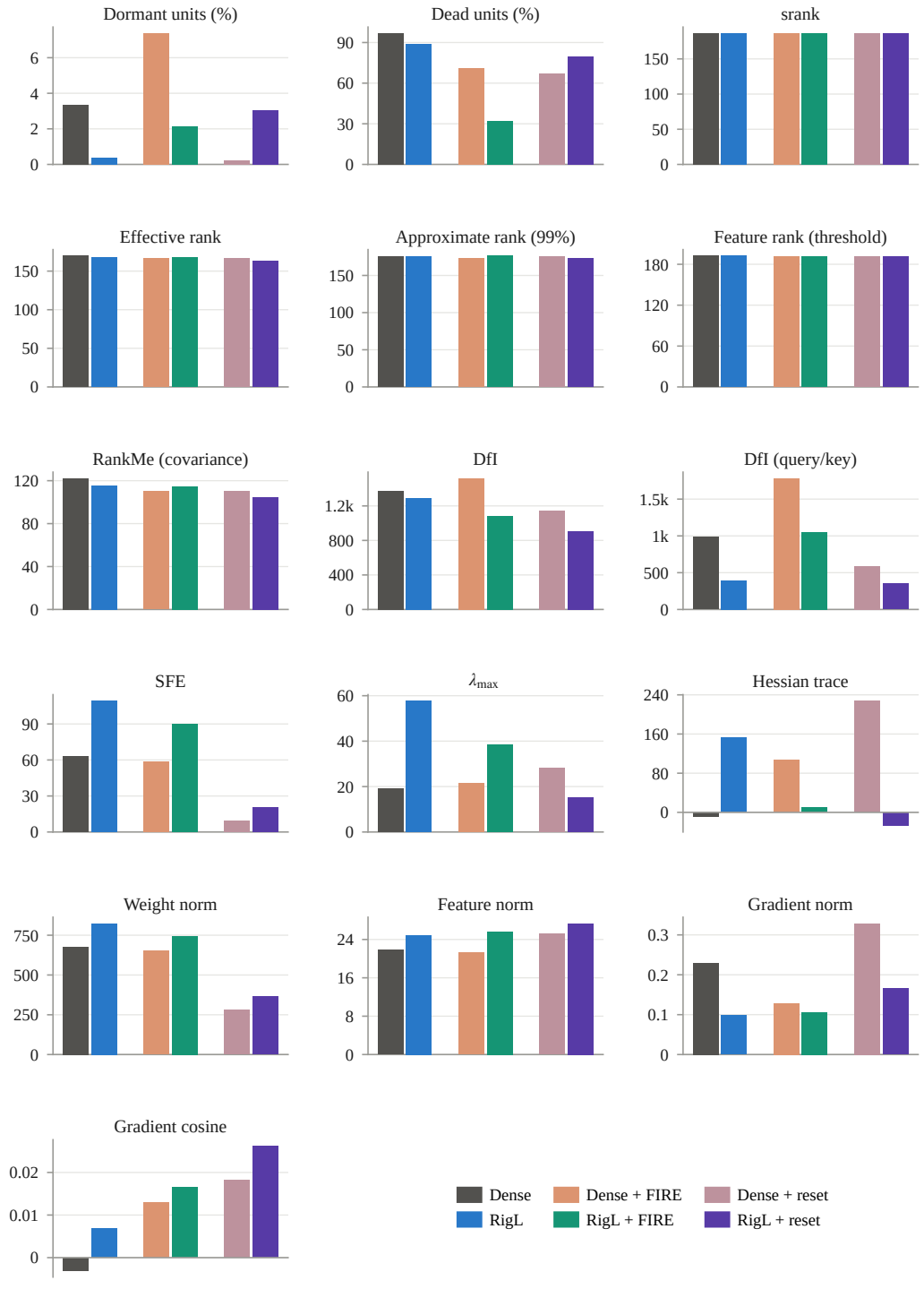


Figure 4: **Diagnostics at the end of training, CIFAR-100/ViT-Tiny, sample-incremental.** Dense, RigL, and their FIRE and full-reset combinations, measured at epoch 90 of the final stage on a fixed probe set; $\lambda_{\max}$ and the Hessian trace are measured at the start of the final stage. Definitions are given in Appendix B.

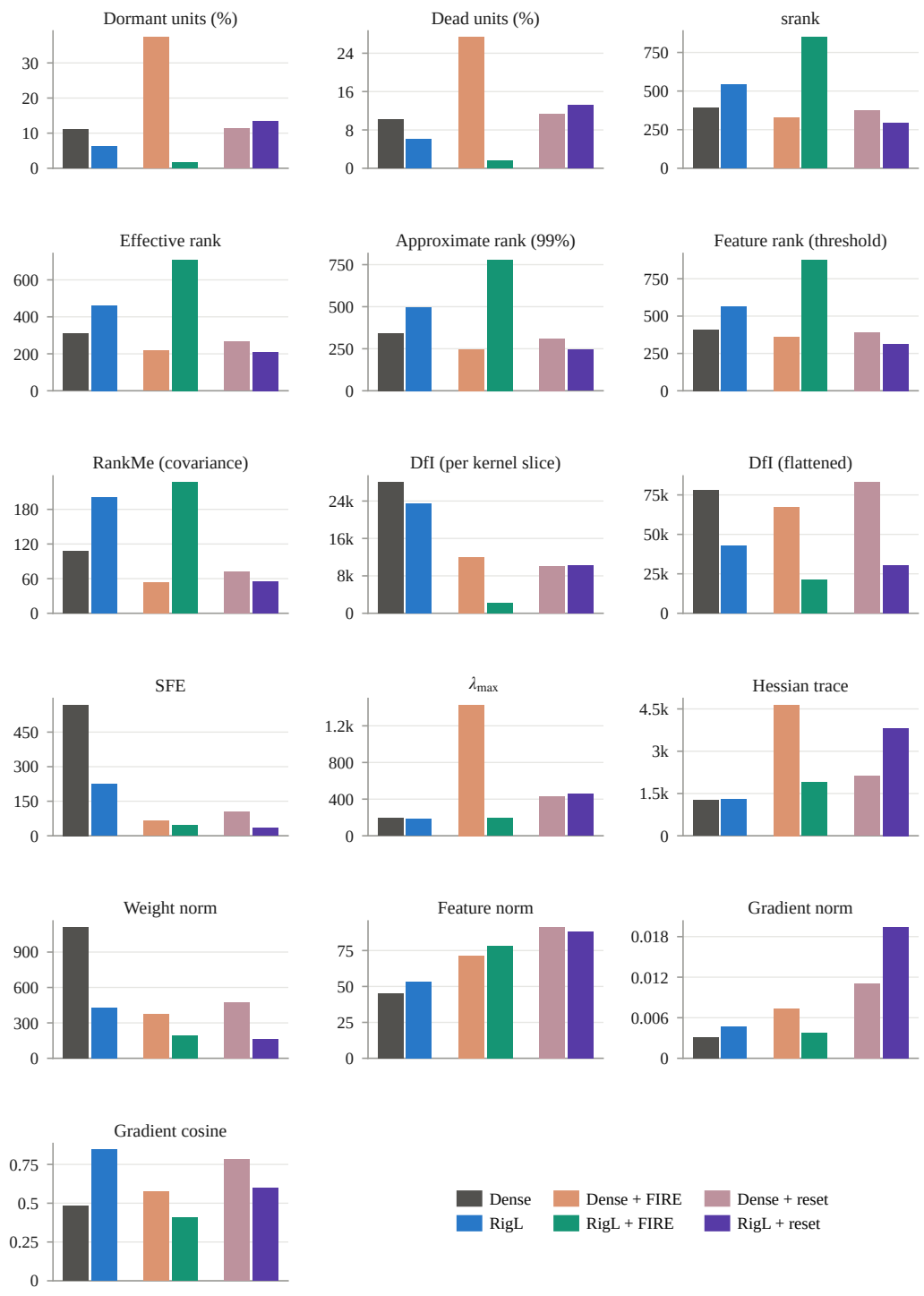


Figure 5: **Diagnostics at the end of training, Tiny ImageNet/VGG-16, sample-incremental.** Arms and measurement points as in Figure 4. DfI is reported per kernel slice and on flattened kernels.

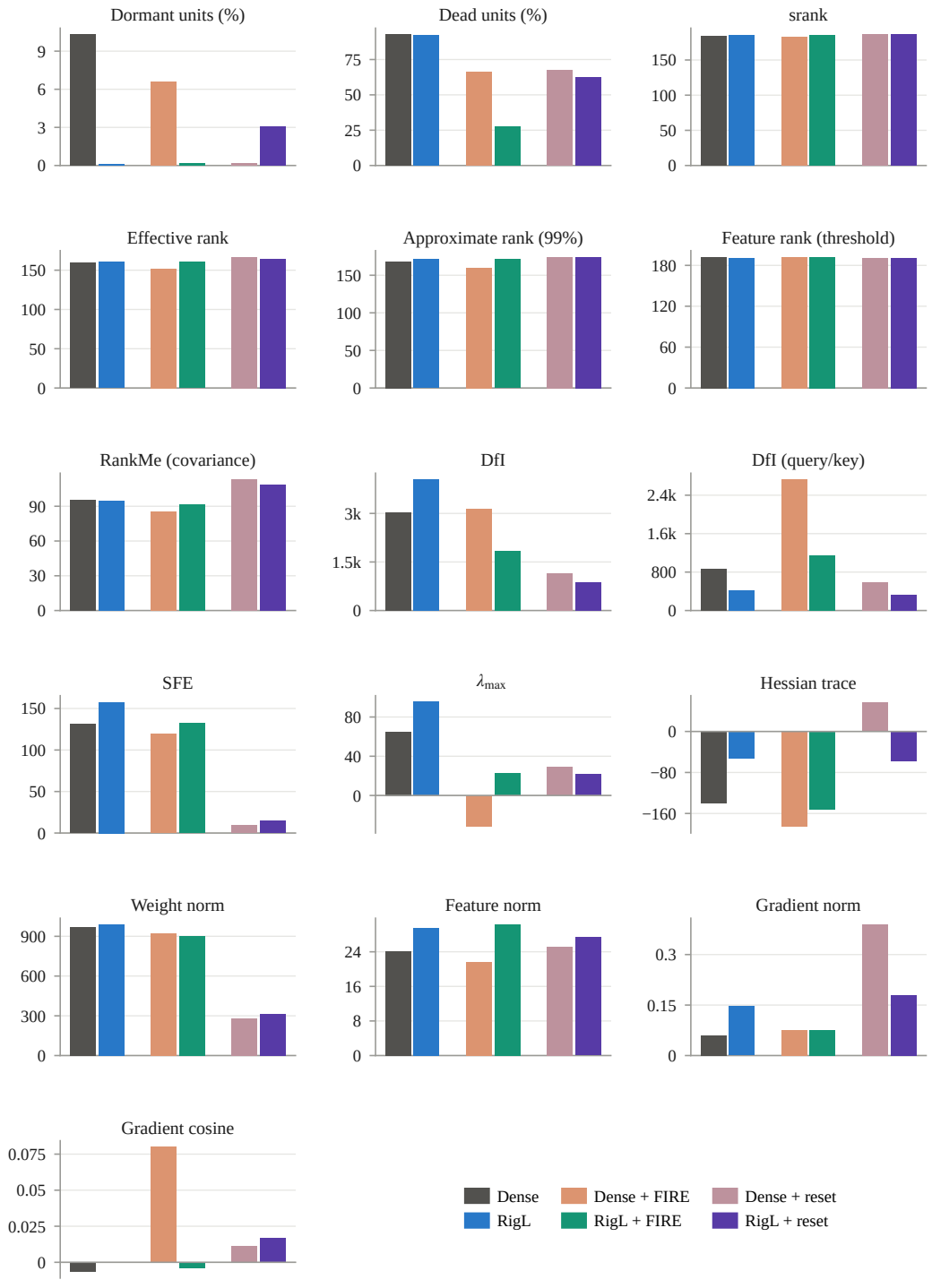


Figure 6: **Diagnostics at the end of training, CIFAR-100/ViT-Tiny, class-incremental.** Arms and measurement points as in Figure 4, after twenty stages.

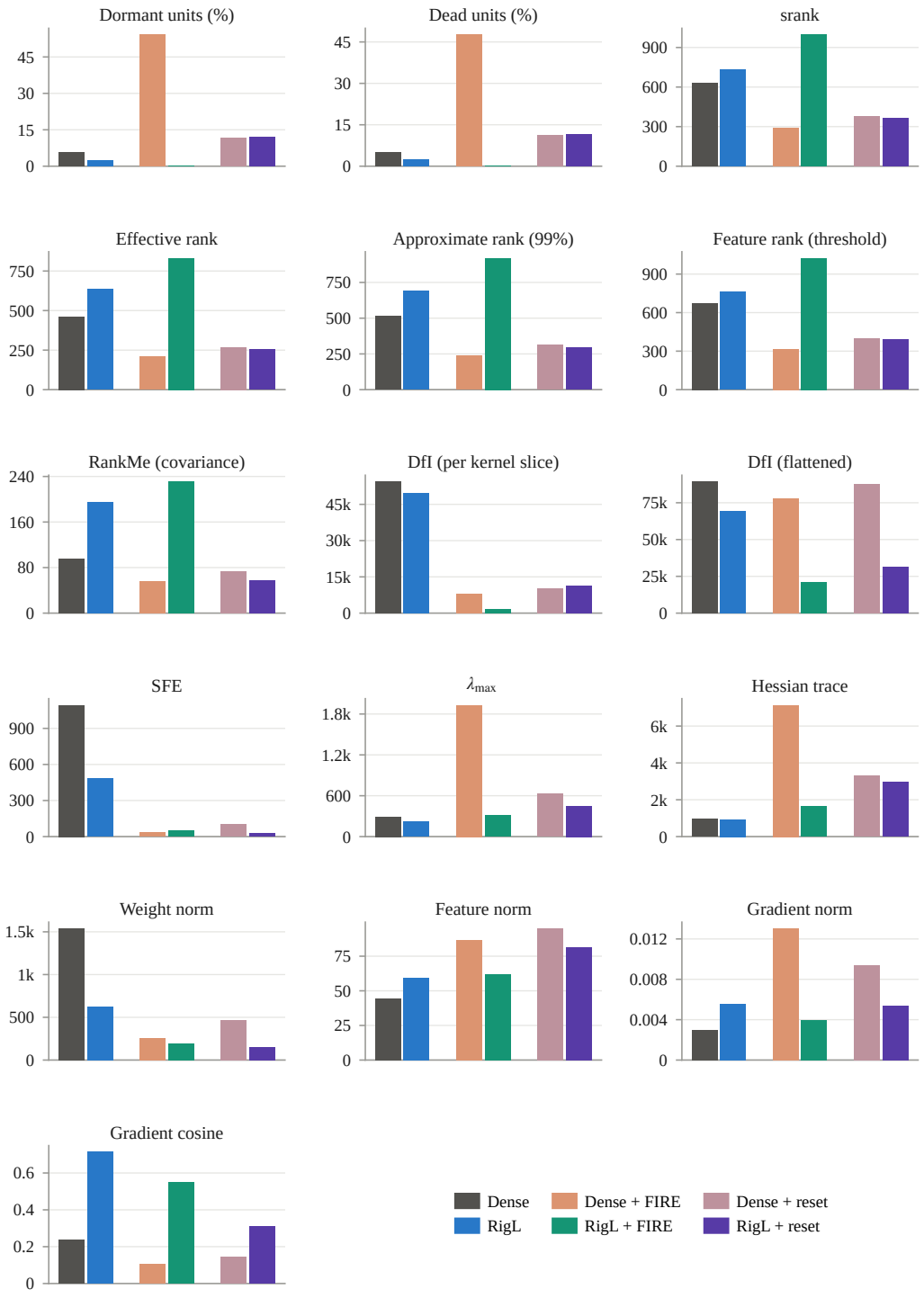


Figure 7: **Diagnostics at the end of training, Tiny ImageNet/VGG-16, class-incremental.** Arms and measurement points as in Figure 5, after twenty stages.

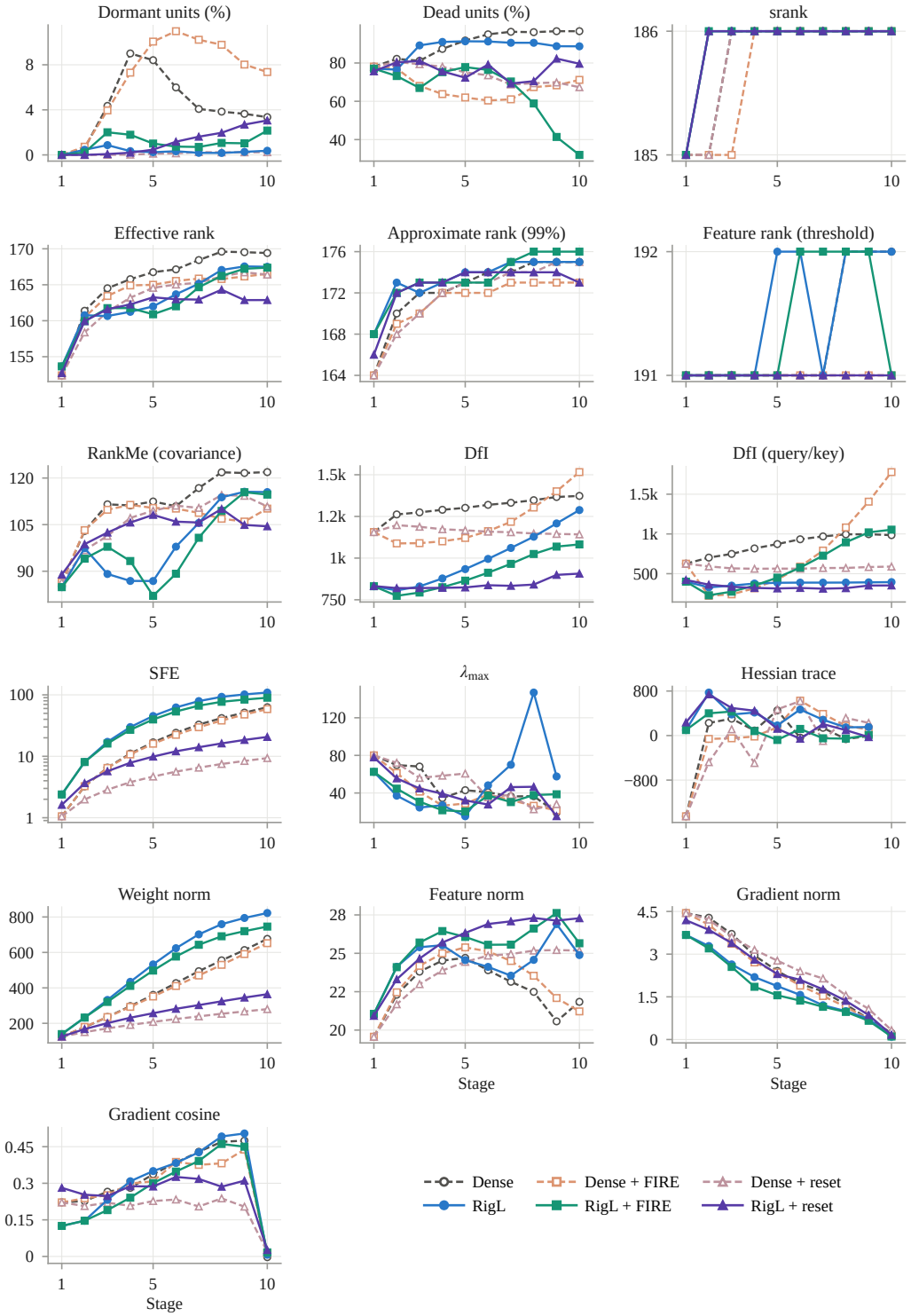


Figure 8: **Diagnostics over training, CIFAR-100/ViT-Tiny, sample-incremental.** The quantities of Figure 4 at the end of every stage. Dashed lines with hollow markers denote dense training and solid lines with filled markers denote RigL; $\lambda_{\max}$ and the Hessian trace are measured at the start of the next stage and end one stage early.

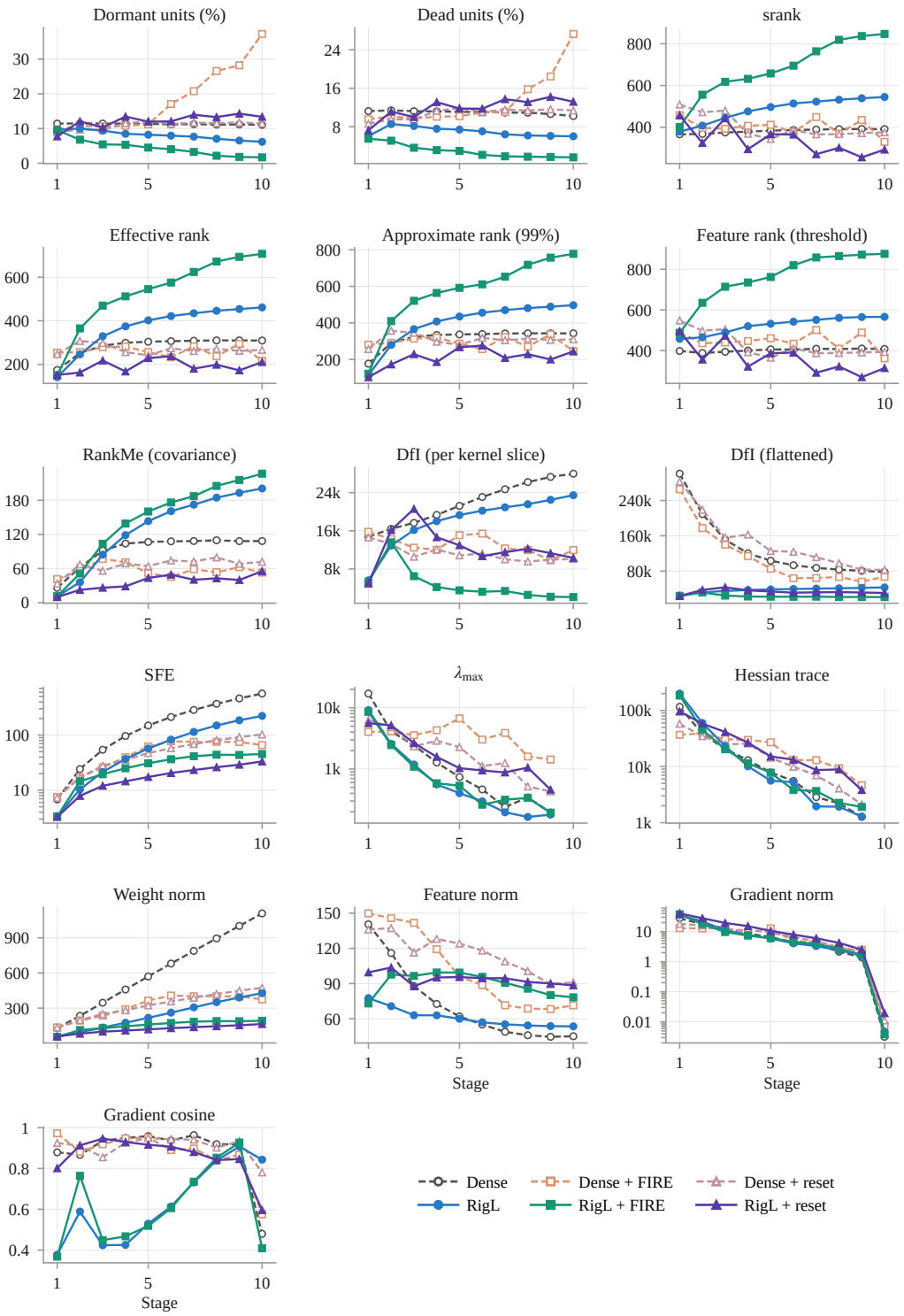


Figure 9: **Diagnostics over training, Tiny ImageNet/VGG-16, sample-incremental.** The quantities of Figure 5 at the end of every stage, drawn as in Figure 8.

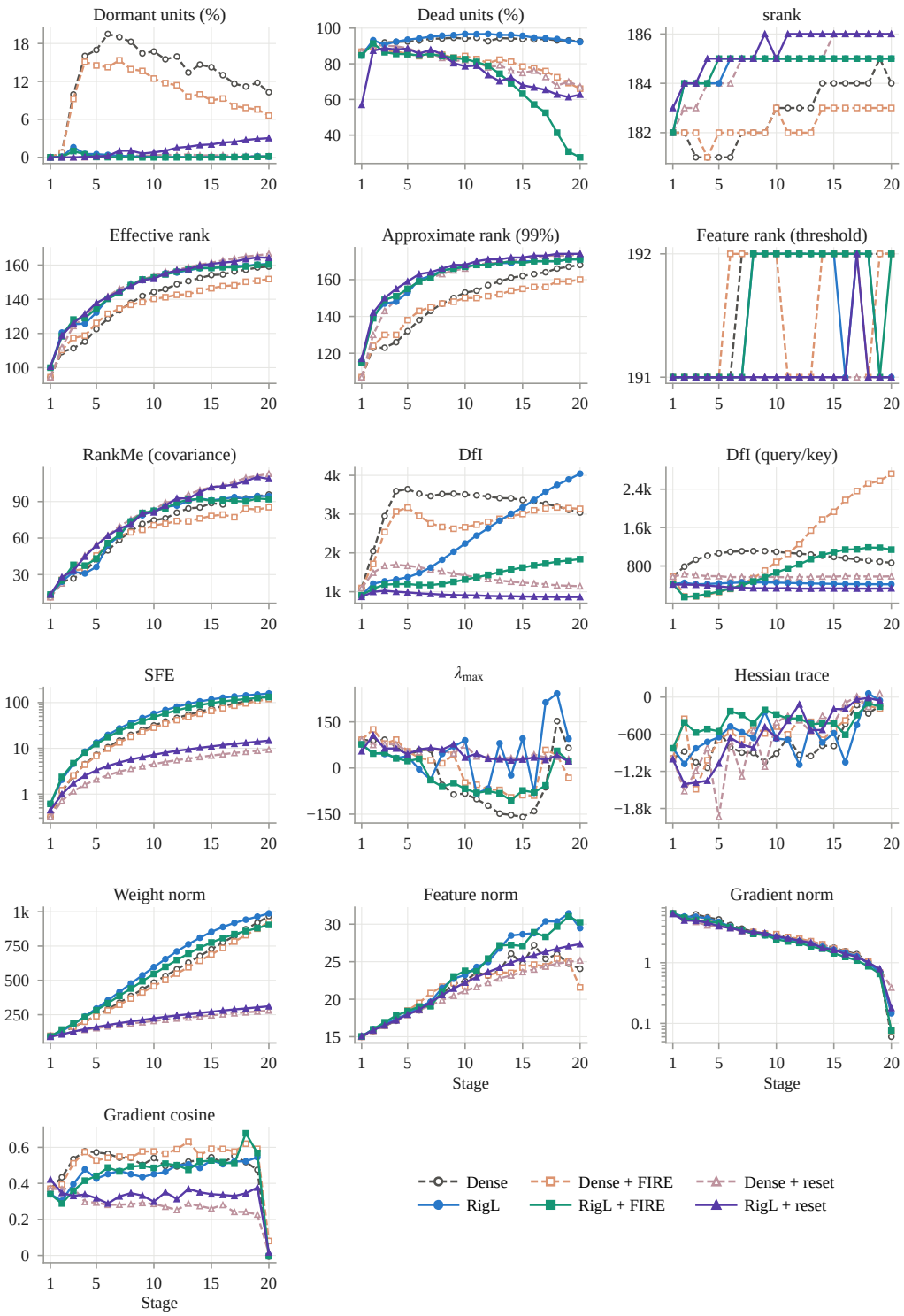


Figure 10: **Diagnostics over training, CIFAR-100/ViT-Tiny, class-incremental.** As Figure 8, over twenty stages.

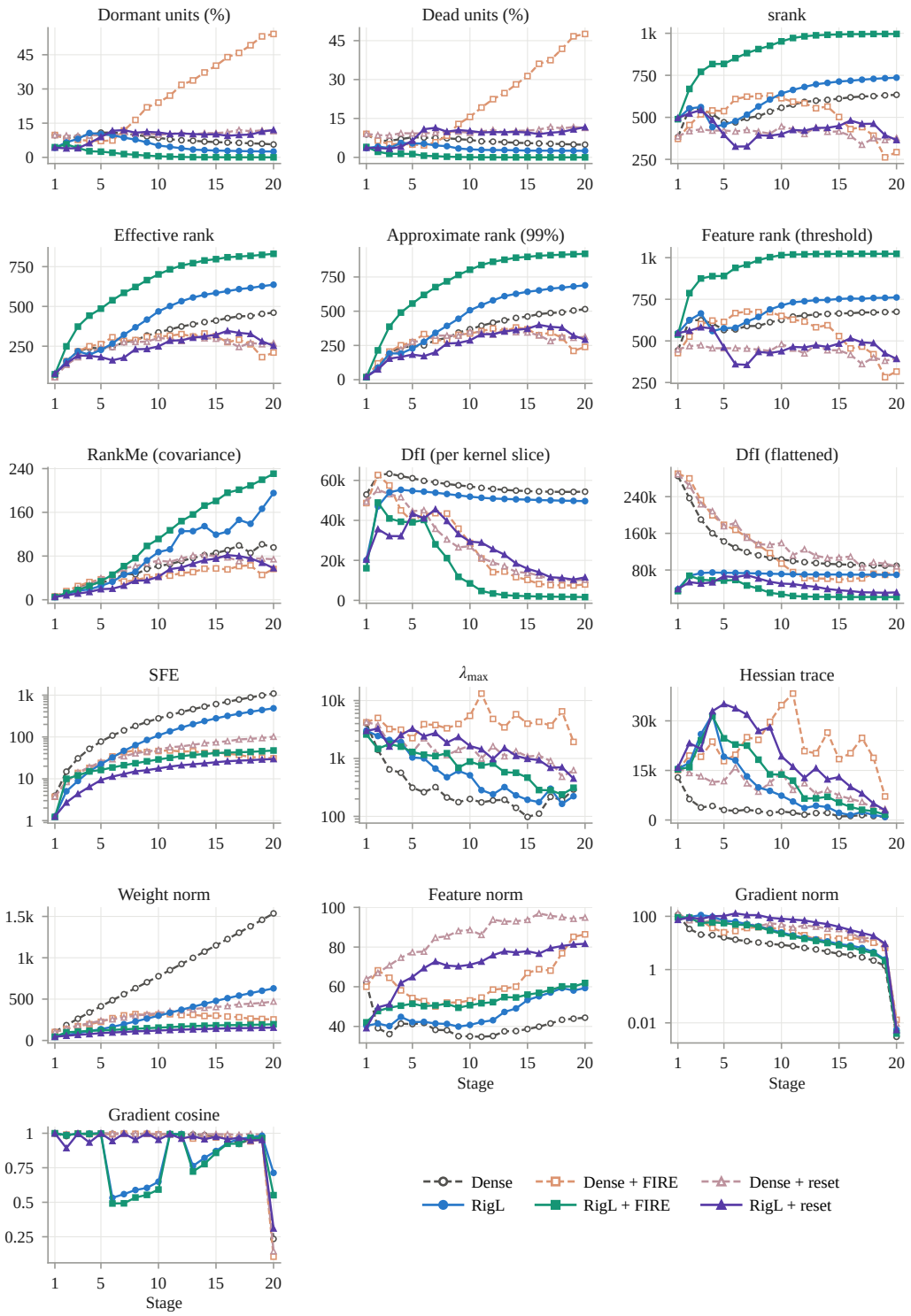


Figure 11: **Diagnostics over training, Tiny ImageNet/VGG-16, class-incremental.** As Figure 9, over twenty stages.

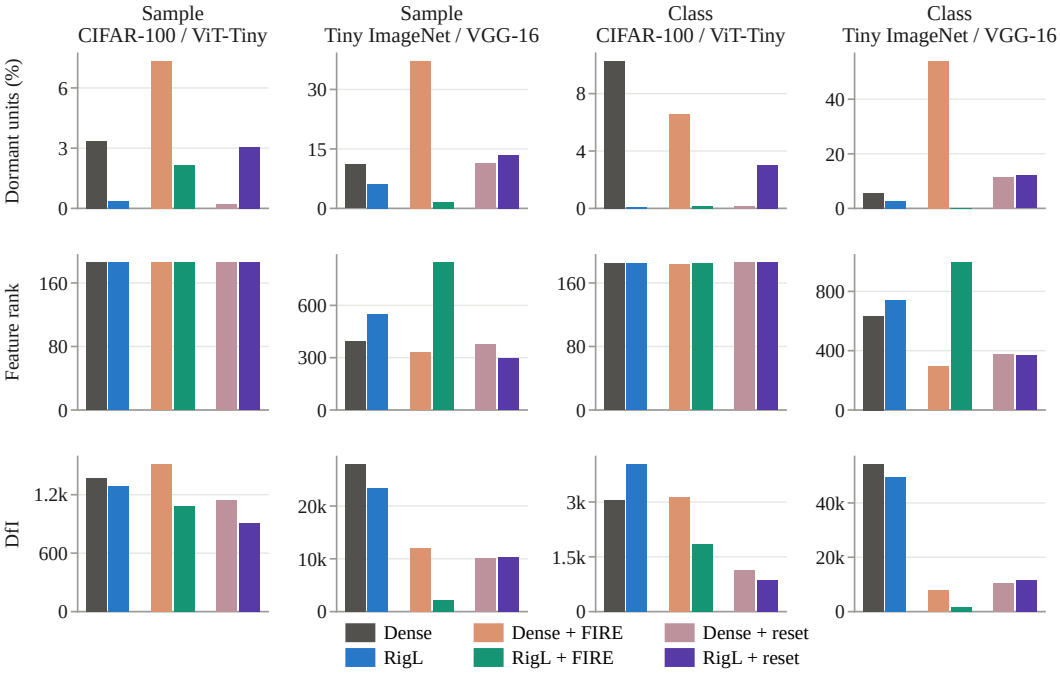


Figure 12: **Focused plasticity diagnostics.** Dense, RigL, and their FIRE and full-reset combinations at the end of training, measured on a fixed probe set. Definitions in Appendix B.

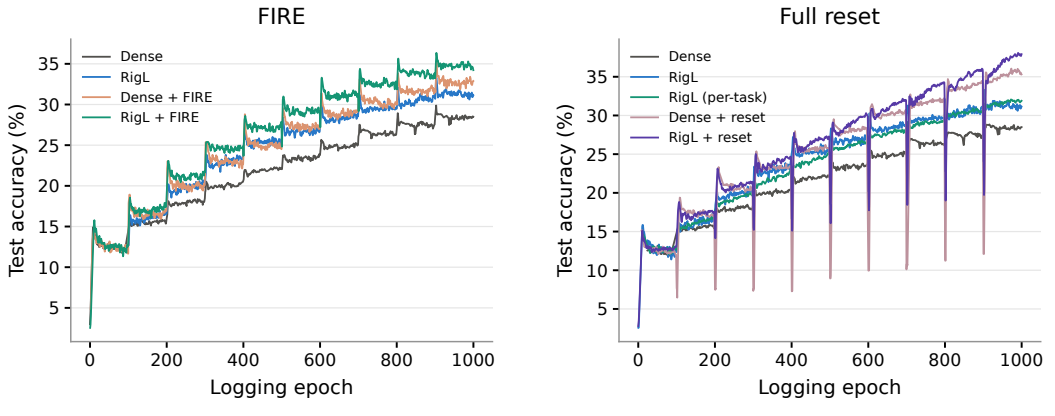


Figure 13: **Preliminary intervention trajectories.** Curves recovered approximately from raster color traces. Overlapping curves and steep boundary changes limit precision; source uncertainty bands are not reconstructed. Endpoint statements in the main text use the source's printed values instead.

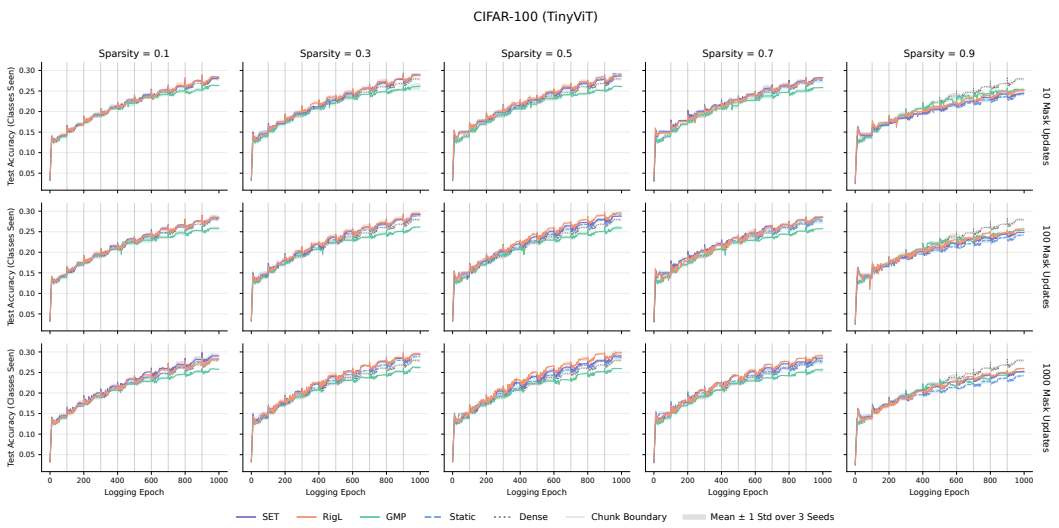


Figure 14: **Incomplete sweep reconstruction.** CIFAR-100/ViT-Tiny, sample-incremental, drop fraction 0.3. Columns vary sparsity; rows vary cadence parameter U.

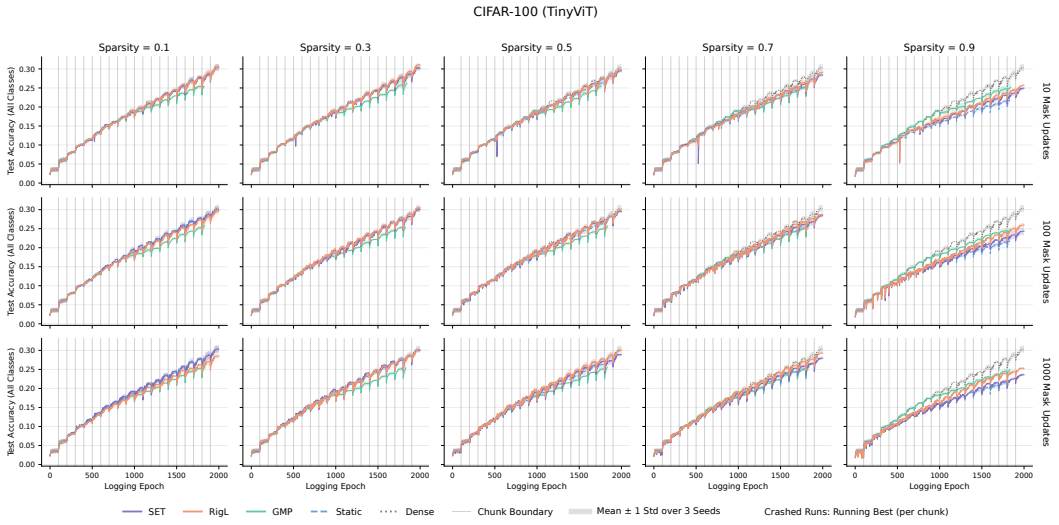


Figure 15: **Incomplete sweep reconstruction.** CIFAR-100/ViT-Tiny, sample-incremental, drop fraction 0.7. Columns vary sparsity; rows vary cadence parameter U. Missing curves are not imputed. Source bands and any running-best preprocessing are retained.

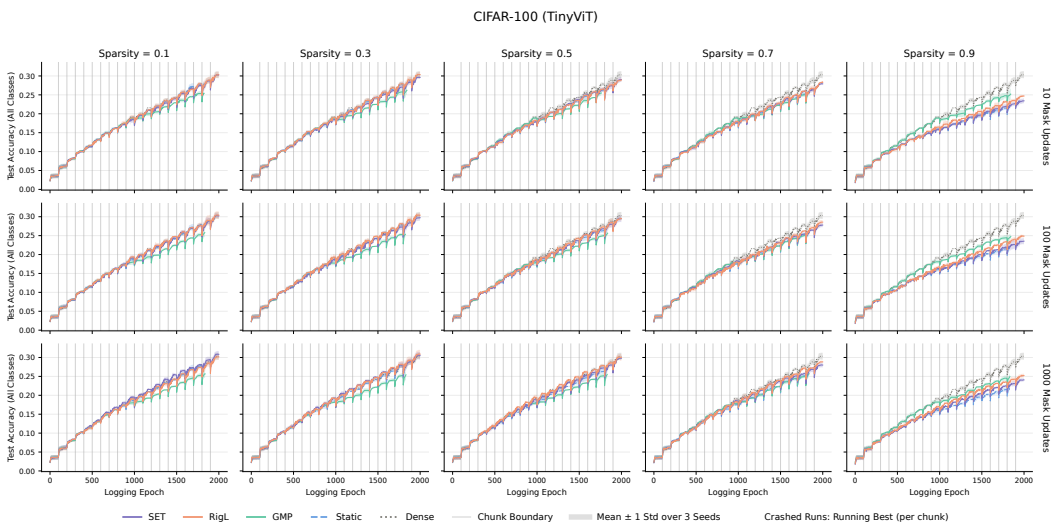


Figure 16: **Incomplete sweep reconstruction.** CIFAR-100/ViT-Tiny, class-incremental, all-class accuracy, drop fraction 0.3. Columns vary sparsity; rows vary cadence parameter U. Missing curves are not imputed. Source bands and any running-best preprocessing are retained.

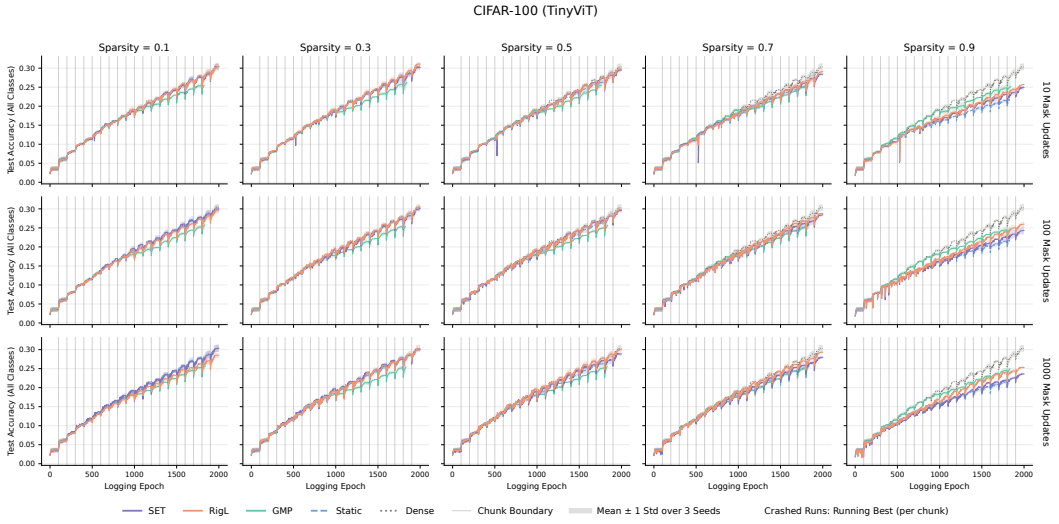


Figure 17: **Incomplete sweep reconstruction.** CIFAR-100/ViT-Tiny, class-incremental, all-class accuracy, drop fraction 0.7. Columns vary sparsity; rows vary cadence parameter U. Missing curves are not imputed. Source bands and any running-best preprocessing are retained.

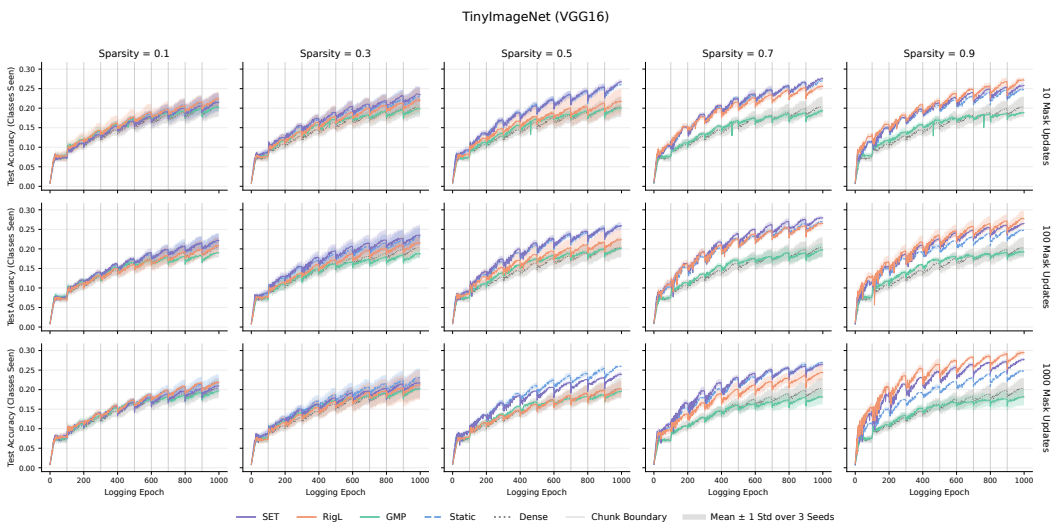


Figure 18: **Incomplete sweep reconstruction.** Tiny ImageNet/VGG-16, sample-incremental, drop fraction 0.3. Columns vary sparsity; rows vary cadence parameter U. Missing curves are not imputed. Source bands and any running-best preprocessing are retained.

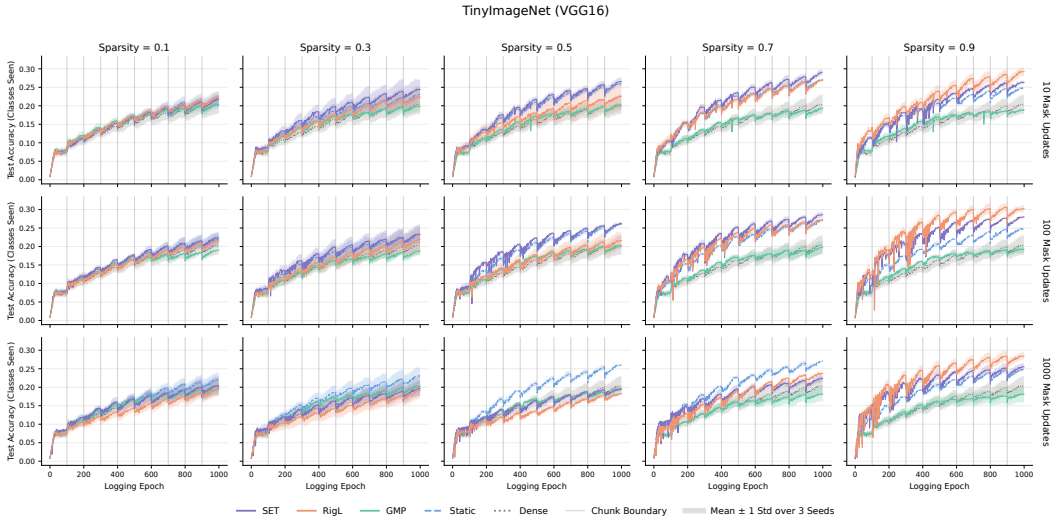


Figure 19: **Incomplete sweep reconstruction.** Tiny ImageNet/VGG-16, sample-incremental, drop fraction 0.7. Columns vary sparsity; rows vary cadence parameter U. Missing curves are not imputed. Source bands and any running-best preprocessing are retained.