AC/DC on a Budget – Alternating Sparse Phases

Rahul Nittala[1], **Advait Gadhikar**[1], **Tom Jacobs**[1], **Rebekka Burkholz**[1]
[1] CISPA Helmholtz Center for Information Security, Germany
rahul.nittala@cispa.de, advait.gadhikar@cispa.de, tom.jacobs@cispa.de,
burkholz@cispa.de

Abstract

State-of-the-art neural network sparsification relies on alternating dense and sparse training phases. While computationally expensive, dense phases are critical for performance. Our empirical analysis reveals the mechanics behind this gain: upon reintroduction, previously masked parameters receive disproportionately large gradients that drive essential mask rewiring and parameter sign-flipping. Building on these insights, we optimize the performance-to-FLOPs Pareto frontier through three innovations: (1) zero-initializing masked parameters during dense phases to promote relevant sign flips, (2) shortening dense phases, and (3) replacing the dense phase entirely with a sparse rewiring phase that updates only the active mask and a small, high-gradient fraction of out-of-mask weights. Evaluated on CIFAR100 and ImageNet, our variant ARC (Alternating Rewiring Compression) matches baseline AC/DC performance at a fraction of the computational cost, establishing a superior performance-to-FLOPs frontier also compared to standard sparse-to-sparse algorithms. Experiments with GPT-2 on OpenWebText further show improvements in validation perplexity and average zero-shot performance.

1 Introduction

The paradigm of overparameterization has led to significant breakthroughs across various machine learning tasks by enhancing optimization landscapes and model generalization ([51], [65], [2], [3]). Nevertheless, the high computational costs associated with training and deploying these models have increased the need for sparsification methods to alleviate hardware constraints ([45], [8]). However, most state-of-the-art sparsification methods, ranging from iterative magnitude pruning [24, 63] to continuous reparameterization [38, 80], rely on the dense model to guide the search for a generalizable sparse structure. While Pruning at Initialization (PaI) attempts to bypass this by training sparse masks from scratch, it often fails to match the performance of its dense-to-sparse counterparts [17]. This suggests that the initial dense training phase—or the analogous exploration in Dynamic Sparse Training (DST) [49, 13] through dense gradients—provides a critical "alignment" or "support expansion" that is currently irreplaceable. In this paper, we seek to understand the precise role dense training plays in finding optimal sparse masks and ask: Can this computational cost be minimized without sacrificing the benefits of dense alignment?

To investigate this, we use the AC/DC algorithm (Alternating Compressed/DeCompressed training) [60] as our primary lens. AC/DC's explicit alternation between dense and sparse phases allows us to isolate and diagnose the impact of dense updates. We frame this investigation through the theoretical lens of saddle-to-saddle dynamics [58, 77]. Under this view, training is a sequence of transitions between singular modes; we argue that dense phases act as "rewiring windows" that provide access to off-support directions necessary to escape current sparse saddles. Moreover, we characterize this mechanism empirically by measuring parameter dynamics—specifically sign-flips and activation subspace stability. We find that dense phases facilitate critical alignment by allowing weights with small magnitudes (typically masked) but large gradients to re-enter the mask and correct their

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

directional signal [19, 79]. Building on these insights, we make three propositions: (i) zero-initializing masked parameters during dense phases, (ii) truncating dense phases, (iii) Alternating Rewiring Compression (ARC), a method that replaces full dense updates with sparse updates (active sparse mask and a fraction of out of mask weights with large gradients). These three modifications each target a different aspect of the dense phase cost, while preserving the support-expansion mechanism we identify as essential. Our contributions are as follows:

- We propose a theory-motivated view of alternating sparse training as a sequence of sparse consolidation followed by support expansion, which links compressed-gradient Iterative Hard Thresholding (IHT) and saddle-to-saddle dynamics.
- We empirically show that sparse phases refine the current support, while dense phases expose high-gradient off-support coordinates that drive mask rewiring and sign corrections.
- We introduce three innovations: Zero init, truncated dense phases, and the method ARC, which reduces the dense-phase cost by updating only a small high-gradient subset of off-support parameters.
- We demonstrate on CIFAR100 and ImageNet that ARC matches or exceeds the performance of AC/DC while significantly reducing theoretical FLOPs, improving the performance-to-FLOP tradeoff compared to AC/DC and RigL. GPT-2 experiments on OpenWebText additionally demonstrate improved perplexity and average zero-shot performance.

1.1 Related work

Existence of sparse solutions. The feasibility of pruning neural networks has been popularized by the lottery ticket [16], which has also been proven theoretically [47, 57, 53, 7, 6, 10, 20, 15]. Related experimental studies and theoretical investigations suggest that iterative approaches that learn the sparse structure and corresponding weights simultaneously might be required [56, 18].

Sparse training. Sparsification methods aim to find a sparse network that can generalize well with a subset of active parameters. This strategy description covers a broad class of algorithms. Pruning methods like Iterative Magnitude Pruning [16, 24], Learning Rate Rewinding [63], Woodfisher [67], CAP [39], Optimal Brain Surgeon [25, 74] start training a dense network and gradually sparsify it during the course of training, eliminating parameters after scoring them based on magnitude or other importance metrics. The post-training and iterative dense-to-sparse methods benefit from dense optimization and gradual support refinement, but retain substantial dense training cost. Continuous sparsification methods like spred [80], PILoT [27], STR[38], L0 regularized training [46] reparameterize training or use a regularizer to transition from dense to sparse networks. Even deeper reparameterization have been utilized for unstructured and structured sparse training [36, 35]. An orthogonal line of work, dynamic sparse training (DST) trains a random sparse network instead of a dense one. During training, DST methods smartly explore the space of the dense network by rewiring the mask randomly [49], based on gradient information [13, 40] or graph connectivity [78]. As DST methods instead keep the model sparse while periodically changing its connectivity, it can reduce theoretical training FLOPs but makes support discovery more constrained.

AC/DC [60] lies between dense-to-sparse and sparse-training approaches: it alternates fully dense exploration with fixed-support sparse consolidation. This provides access to dense optimization for support discovery, but incurs the dense FLOPs cost at every decompression. ARC retains the dense warmup and alternating support refinement, while replacing subsequent dense phases with sparser expansions of the inherited sparse support selected based on gradients. Our focus is therefore on retaining the support-expansion benefit of AC/DC without the fully dense intermediate training phases. Top-KAST similarly explores parameters beyond its sparse forward support, using a larger backward support, both selected by weight magnitude [32]. In contrast, ARC selects additional parameters based on off-support gradients at rewiring-phase boundaries and trains the expanded support in both forward and backward passes before pruning back to the target sparsity.

Training a fixed sparse mask from scratch. Pruning at initialization (PaI) selects a sparse mask before training and keeps its connectivity fixed [44, 20, 68, 42, 71, 61, 55]. Its difficulty in matching dense-to-sparse methods [17] motivates understanding what dense training contributes to support discovery. AC/DC's explicitly separated phases allow us to isolate this contribution.

Small weights and sign flip importance. Recent works in sparse training [18, 19] have highlighted sign flipping as a primary performance bottleneck, enabling relevant sign-flips improves sparse training and can also improve optimization [28]. For this reason we define and measure these consequential sign-flips in the context of AC/DC to investigate parameter dynamics in dense training. On top of that to encourage sign flips we can focus on small weight parameters in the dense/rewiring phase [79].

Implicit bias and saddle-to-saddle dynamics. Implicit bias studies how optimization, architecture, and parameterization select particular solutions in overparameterized models beyond what is specified by the training objective alone [23, 73, 59, 27, 29, 43, 30, 12, 72, 14, 70]. More specifically, we draw on the saddle-to-saddle view of learning dynamics, a key driver behind implicit regularization. [66] derive exact gradient-flow dynamics for deep linear networks and show that learning proceeds progressively across singular modes, typically prioritizing dominant modes first in line with [1, 34, 41]. Building on this type of learning description, [31] shows small-initialization training as a sequence of transitions between saddle regions, where symmetry and sparsity influence which directions become active. For deep diagonal linear networks, [58] characterize saddle-to-saddle dynamics exactly in which coordinates are activated progressively along the optimization path or deactivated. Finally, [77] connect such transitions to simplicity bias in deep neural networks. This perspective is particularly relevant to alternating sparse training in general: fixed sparse phases can be interpreted as consolidating the currently active representation, while dense or rewiring phases provide access to new off-support directions that enable the next transition. We use this viewpoint as conceptual motivation for studying mask rewiring and support expansion in AC/DC.

2 Theory: Alternating sparse training phases

Our central question is whether the benefits of dense support expansion can be retained without updating every parameter. We first extend the convergence analysis of AC/DC to compressed gradients, then examine how alternating sparse phases can preserve saddle-to-saddle learning dynamics.

AC/DC (Alternating Compressed/DeCompressed Training) [60] follows an iterative hard-thresholding (IHT) principle [5] to minimize $f : \mathbb{R}^N \to \mathbb{R}$ subject to $\|\theta\|_0 \leq k$, where k is the target number of nonzero parameters. After dense warmup, AC/DC alternates magnitude-pruned, fixed-support sparse training with dense phases that restore masked parameters to their values at the end of the previous dense phase (*Stale init*), followed by sparse finetuning (Figure 6).

Notation and compressed updates. For a vector v, $\mathrm{Top}_s(v)$ returns the indices of its s largest-magnitude entries, with ties resolved by a fixed rule. For a coordinate set A, P_A retains coordinates in A and zeros the others, while v_A denotes restriction to A. The hard-thresholding operator is $T_s(v) = P_{\mathrm{Top}_s(v)}v$.

At iteration t, let θ_t be the current parameters, $S_t = \mathrm{supp}(\theta_t)$ their sparse support, and g_t a stochastic gradient. We consider the expanded support $A_t = S_t \cup G_t$, where $G_t = \mathrm{Top}_q((g_t)_{S_t^c})$ selects the q largest-magnitude gradient coordinates outside S_t. The compressed gradient is $h_t = P_{A_t} g_t$. The inherited support preserves the current sparse representation, while G_t provides additional directions for support expansion.

Convergence with compressed gradients. We adapt the stochastic-IHT convergence analysis underlying AC/DC by replacing the dense gradient with h_t. As in the original analysis, we assume restricted smoothness, which controls objective changes along sparse directions, and a concentrated Polyak–Łojasiewicz (CPL) condition, which requires sufficiently large gradient energy to be concentrated on a small set of coordinates. For a global minimizer $\theta^\star$, the (s, α)-CPL condition requires

$$\|T_s(\nabla f(\theta))\|_2^2 \geq 2\alpha\big(f(\theta) - f(\theta^\star)\big). \tag{1}$$

The precise assumptions are provided in Appendix B.

Replacing dense gradients with compressed updates additionally requires that the discarded gradient information be sufficiently small on the coordinates relevant to the IHT analysis.

Assumption 2.1 (Compressed-gradient accuracy)**.** *Let $\Delta_t = f(\theta_t) - f(\theta^\star)$. For every support U_t arising in the IHT analysis with $|U_t| \le 2k + q + 3k^\star$, the compressed gradient satisfies $\mathbb{E}[\|(h_t - \nabla f(\theta_t))_{U_t}\|_2^2 \mid \theta_t] \le \sigma_{\mathrm{cg}}^2 + \rho\Delta_t$, for constants $\sigma_{\mathrm{cg}}^2, \rho \ge 0$.*

Here σ_{cg}^2 bounds the error independent of suboptimality, while ρ controls the state-dependent error introduced by gradient compression.

Theorem 2.2 (Convergence with compressed gradients)**.** *Let $f : \mathbb{R}^N \to \mathbb{R}$ have a $k^\star$-sparse global minimizer $\theta^\star$. Suppose $\beta > \alpha > 0$, f is $(2k + q + 3k^\star, \beta)$-restricted smooth and $(k^\star, \alpha)$-CPL, and Assumption 2.1 holds. For universal constants $C_0, C_1 > 0$, let $k \ge C_0 k^\star(\beta/\alpha)^2$ and define $\lambda_{\mathrm{cg}} = \alpha/(32\beta) - C_1\beta\rho$. If $\lambda_{\mathrm{cg}} > 0$, stochastic IHT with $\eta = 1/(2\beta)$ and update $\theta_{t+1} = T_k(\theta_t - \eta h_t)$ converges in $O(\lambda_{\mathrm{cg}}^{-1}\log(\Delta_0/\varepsilon))$ iterations to an expected suboptimality of at most $\varepsilon + O(\beta\sigma_{\mathrm{cg}}^2/\lambda_{\mathrm{cg}})$.*

The proof is provided in Appendix B. Thus, compressed-gradient rewiring preserves the qualitative IHT convergence behavior when the expanded support captures sufficient dense-gradient information, with compression error reducing the effective contraction rate.

Alternating sparse phases can preserve saddle-to-saddle learning dynamics. Beyond convergence, we seek to understand why the coordinates needed for learning change throughout training. Saddle-to-saddle dynamics describe learning as successive transitions between singular modes [58, 77]. In deep linear networks, learning progressively increases effective rank, prioritizing dominant singular directions [1, 34, 41]. We argue that alternating sparse phases can support the same progression: sparse phases consolidate previously learned modes, while rewiring phases expand the support to activate new ones.

We consider a linear network with one hidden layer, $f_{W_1,W_2}(x) = W_2W_1x$ [66], trained with whitened inputs and population loss $\mathcal{L}(W_1, W_2) = \frac{1}{2}\|W_2W_1 - M\|_F^2$. Here $M = \mathbb{E}[yx^\top] = U\,\mathrm{diag}(s_1, \ldots, s_r)V^\top$, where $s_1 > \cdots > s_r > 0$ are the singular values, with corresponding singular vectors u_i, v_i. Write $\theta = (\mathrm{vec}(W_1), \mathrm{vec}(W_2))$ and $g(\theta) = \nabla_\theta\mathcal{L}(\theta)$. Let $M_m = \sum_{i=1}^m s_iu_iv_i^\top$ denote the first m singular modes and $\mathcal{M}_m$ the corresponding rank-m saddle configurations.

We use m to index successive mode-learning phases. At phase m, let θ_m denote the parameters at the phase boundary, S_m the inherited sparse support, and $A_m = S_m \cup G_m$ its expansion, where $G_m = \mathrm{Top}_q((g(\theta_m))_{S_m^c})$. Let $e_{m+1}(\theta)$ denote the dense gradient component associated with activating the next singular mode.

The result relies on four assumptions: previously learned modes admit sparse representations preserved by pruning; the residual near $\mathcal{M}_m$ is dominated by the next singular mode; the inherited support captures little of the corresponding escape direction; and the expanded support captures most of it. We denote the sparse approximation error by δ_m, the fraction of the escape direction captured by S_m by α_m, and the fraction missed by A_m by ε_m. The formal conditions are stated in Assumptions C.9–C.12 in Appendix C. The connection to gradient compressibility is given in Proposition C.7.

Theorem 2.3 (Alternating sparse phases implement sparse saddle-to-saddle dynamics)**.** *Suppose Assumptions C.9– C.12 hold for phases $m = 0, \ldots, r-1$, and $\mathcal{L}$ is L-smooth in a neighborhood of the relevant trajectories. Then each alternating cycle has the following behavior.*

1. ***Consolidation.*** *Training on S_m primarily optimizes the sparse representation of previously learned modes $1, \ldots, m$, approaching a neighborhood of the rank-m saddle $\mathcal{M}_m$.*
2. ***Stalling on the old support.*** *The inherited support contains only a small fraction α_m of the escape direction toward mode $m+1$. Hence, training exclusively on S_m leaves the trajectory near the current sparse saddle.*
3. ***Sparse escape by rewiring.*** *The expanded support $A_m = S_m \cup G_m$ captures at least a $1 - \varepsilon_m^2$ fraction of the dense escape-direction energy, satisfying $\|P_{A_m}e_{m+1}(\theta)\|_2^2 \ge (1-\varepsilon_m^2)\|e_{m+1}(\theta)\|_2^2$. Thus, rewiring provides access to directions needed to increase the learned strength of mode $m+1$.*
4. ***Support transition.*** *After rewiring, magnitude pruning retains the coordinates needed to represent the newly learned mode, producing a support S_{m+1} that represents M_{m+1} up to approximation error δ_{m+1}.*

The result connects alternating sparse consolidation and support expansion to progressive mode learning. When the expanded support captures the relevant escape directions, sparse rewiring can preserve the transitions enabled by dense phases without requiring all parameters to be updated. The formal analysis is provided in Appendix C.

3 The Mechanics of AC/DC: Diagnosis

The theory in Section 2 shows that alternating sparse training can be viewed as a sequence of consolidation and support-expansion steps. We next ask whether related phase roles appear in practical AC/DC training: do fixed-support sparse phases mainly refine the current representation, while dense phases expose useful off-support directions? For this, we study AC/DC before introducing our algorithmic modifications. These diagnostics clarify the role of dense phases and indicate which parts of them may be compressible. Unless otherwise stated, we use ResNet18 on CIFAR100; training details are provided in Appendix D.

3.1 Sparse Phases Consolidate While Dense Phases Rewire

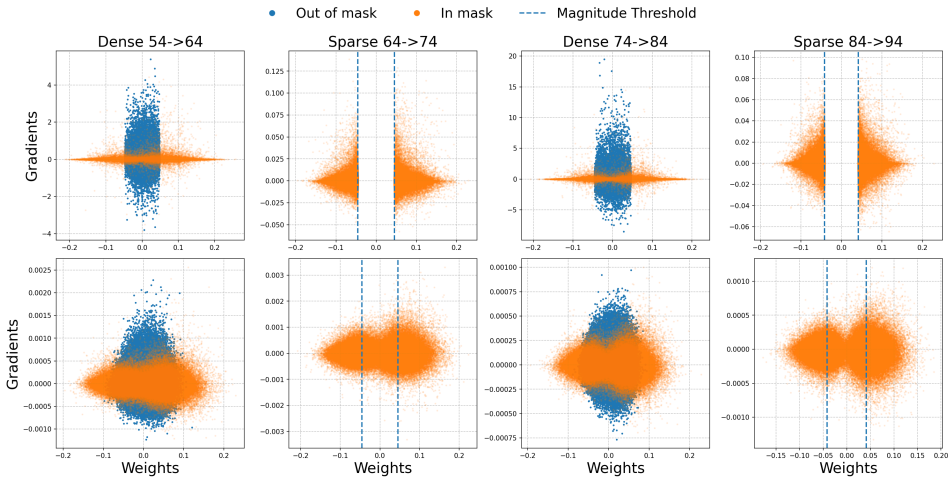


Figure 1: **Gradient–weight dynamics across AC/DC phases.** Gradient versus weight scatter plots for *layer4.0.conv2* of a 90% sparse ResNet18 on CIFAR100. Top and bottom rows show the beginning and end of each phase, respectively. During sparse phases, many active weights move toward small magnitude and small gradient. During dense phases, many reintroduced out-of-mask weights remain near zero but receive large gradients.

We first examine how coordinates retained by magnitude pruning differ from those excluded by the current mask. Figure 1 compares their weights and gradients across consecutive phases, connecting mask membership to the learning signal available to each coordinate.

Sparse phases refine the active support. Sparse training is restricted to the magnitude-pruned support S_t. Within this support, many active weights move toward smaller magnitudes and also have small gradients. Sparse phases therefore consolidate the current representation, separating coordinates that carry useful signal from low-magnitude, low-gradient coordinates that may be removed at the next pruning step.

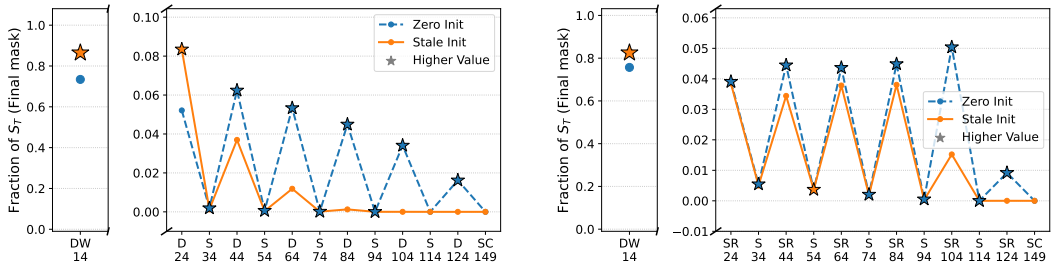
Dense phases expose off-support gradient signal. When masked parameters in S_t^c are reintroduced, many have smaller magnitudes but receive large gradients, forming the vertical band near zero in Figure 1. This contrasts with the small, low-gradient active weights (S_t) at the end of sparse training. Dense phases allow these off-support coordinates to grow and compete for inclusion in the next mask. Thus, the dense phases act as rewiring windows, enabling the support for the next sparse phase to differ from the previous one. This mechanism is in line with the support-expansion role as described by Theorem 2.3.

3.2 Sign flips

The gradient–weight analysis suggests that dense phases facilitate changes in the sparse mask. Yet, prior work has shown that the signs of surviving weights are also important for sparse training, and that improving sign alignment can help close the gap between sparse training from scratch and dense-to-sparse training [18, 19]. We therefore ask which phases of AC/DC establish the eventual signs of the parameters retained in the final sparse model. To study this retrospectively, we record the phase of each final-mask parameter's last sign flip, after which the sign remains consistent across phases. Parameters that never flip are assigned to the warmup bin.

Figure 2a shows that most final-mask parameters settle their signs by the end of warmup. Among parameters corrected later, the last sign flips occur predominantly during dense rather than sparse phases. Dense phases therefore do not merely produce transient gradient activity: they provide windows in which parameters retained by the final sparse model resolve their eventual signs.

We compare two initialization strategies for masked parameters at the beginning of each dense phase. *Stale init* restores these parameters to their values at the end of the previous dense phase, whereas *Zero init* resets them to zero. With Stale init, most late sign corrections occur in early dense phases, while Zero init continues to produce corrections in later phases. This suggests that initialization influences when final parameter signs become established, motivating our Zero init modification in Section 4. The detailed counting protocol, layerwise and shortened-phase results, and corresponding ARC analysis are provided in Appendix G.



(a) AC/DC (b) ARC

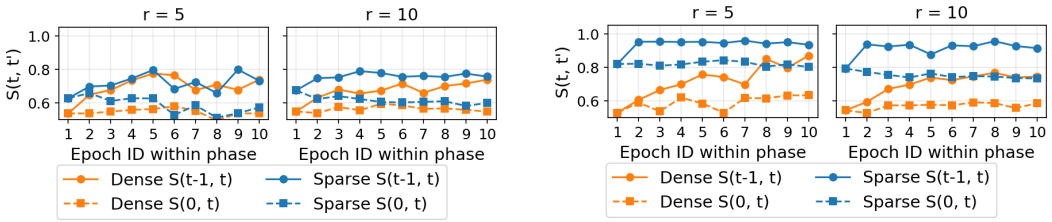
Figure 2: **Timing of final sign alignment.** For each parameter retained in the final sparse mask, we record the phase in which it flips sign for the last time. The y-axis is normalized by the final mask size. DW, D, S, and SC denote dense warmup, dense, sparse, and sparse cooldown phases for AC/DC; in ARC, the corresponding non-sparse phases are rewiring phases on $A_t = S_t \cup G_t$. **(a)** In AC/DC, most final-mask parameters align during dense warmup, while later sign corrections occur mainly during dense phases. **(b)** The analogous ARC diagnostic is discussed after ARC is introduced in Section 4.

3.3 Activation Subspaces Reflect Sparse Consolidation and Dense Rewiring

The preceding diagnostics characterize AC/DC at the level of individual parameters, gradients, and masks. We next ask whether the alternating phases also leave a representation-level signature. The saddle-to-saddle perspective suggests that sparse phases consolidate the current representation, while dense phases allow larger but structured representation changes. We therefore track the leading eigenspaces of activation Gram matrices computed over the CIFAR100 training set at each epoch checkpoint. Their construction is detailed in Appendix F.

Let $U_t^{(r)}$ denote an orthonormal basis for the subspace spanned by the top r activation Gram eigenvectors at epoch t. We measure subspace overlap as $S_r(t_1, t_2) = \frac{1}{r}\left\| {U_{t_1}^{(r)}}^\top U_{t_2}^{(r)} \right\|_F^2$. We report both $S_r(t-1, t)$, measuring consecutive-epoch stability, and $S_r(0, t)$, measuring similarity to the first checkpoint of each phase.

Figure 3 compares an early and a later AC/DC cycle for *layer3.1.conv2*. Sparse phases preserve the leading activation subspaces more strongly than dense phases, especially for $r = 5$ and $r = 10$. This is consistent with sparse phases primarily consolidating the current representation under a fixed



(a) 1st: Sparse 24–34, Dense 34–44

(b) 4th cycle: Sparse 84–94, Dense epochs 94–104

Figure 3: **Sparse phases preserve activation subspaces.** Top-r activation Gram subspace similarity for *layer3.1.conv2* of a 90% sparse ResNet18 on CIFAR100. We show an early and later AC/DC cycle. Sparse phases maintain higher subspace similarity. Dense phases show larger but smooth drift.

support. Dense phases, in contrast, show larger drift from the phase start, indicating more substantial representation updates. Importantly, dense-phase similarity remains substantial and changes smoothly across epochs. Thus, dense phases do not appear to reset the representation, but rather modify it while preserving much of the dominant activation subspace. The distinction between the phases becomes clearer in later cycles, where sparse phases show very high consecutive-epoch similarity while dense phases continue to permit representation changes. Full results across all four cycles and eigendirection heatmaps are provided in Appendix F.

Together, the diagnostics identify dense phases as rewiring windows: they expose high-gradient off-support coordinates, enable persistent sign corrections, and permit smooth representation changes. This motivates preserving the inherited support while replacing full dense optimization with targeted updates of high-gradient off-support coordinates.

4 Method: Alternating Rewiring Compression

Motivated by these diagnostics, we pursue three complementary modifications to improve the performance–FLOP tradeoff: Zero init, truncated dense phases, and *Alternating Rewiring Compression* (ARC).

Zero init. In standard AC/DC, reintroduced parameters inherit their previous dense-phase values (*Stale init*), including their signs. Motivated by the sign-flip diagnostics in Section 3.2, we instead consider *Zero init*, which resets these parameters to zero: $\theta_i \leftarrow 0$ for $i \in S_t^c$. This removes inherited sign information, allowing the current gradient to determine their initial direction before they compete for inclusion in the next sparse mask. Zero init applies to all masked parameters during AC/DC's dense phases, and only to the selected rewiring coordinates G_t in ARC.

Truncated dense phases. Standard AC/DC uses dense and sparse phases of equal length. If dense phases serve as rewiring windows, an immediate practical question is how long those windows need to be. We therefore evaluate schedules that keep the total cycle length fixed but shorten the dense portion. If a standard cycle uses D dense epochs and S sparse epochs, we replace it with $D' < D$ dense epochs and $S' = D + S - D'$ sparse epochs. This preserves the alternating schedule while reducing theoretical FLOPs by replacing dense epochs with sparse epochs. Furthermore, it is consistent with the smooth and gradual nature of the representation changes observed in Section 3.3.

ARC: sparse rewiring instead of dense phases. Zero init and truncation reduce the cost or improve the behavior of dense phases, but still rely on dense updates. ARC replaces the dense phase with a *sparse rewiring phase*. Instead of updating all parameters, ARC updates only $A_t = S_t \cup G_t$, with $G_t = \mathrm{Top}_q\big((|g_t|)_{S_t^c}\big)$, where G_t contains the largest-gradient coordinates outside the current sparse support. Here, q is the number of added parameters that were previously masked. In Section 5, the d in label like $ARC(d)$ specifies the density of G_t over the set of all parameters. The gradients used to construct G_t are aggregated over a small number of randomly sampled mini-batches, and G_t is selected once at the beginning of each rewiring phase. See Appendix D for more details.

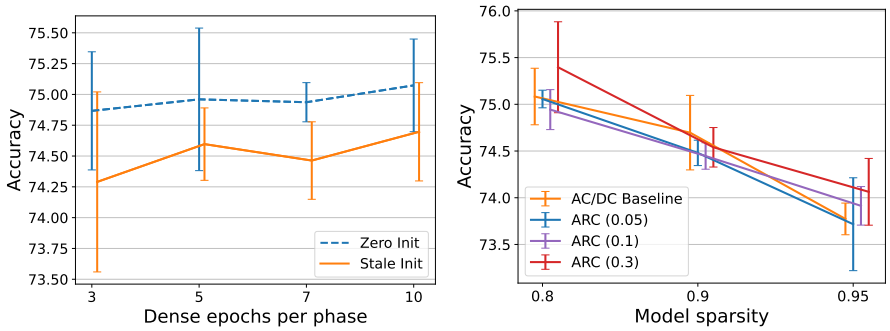


Figure 4: **Reducing and sparsifying rewiring phases.** Results for ResNet18 on CIFAR100. Left: varying the number of dense epochs per AC/DC cycle at 90% sparsity. Right: comparing AC/DC with ARC variants using different G_t budgets across target sparsities.

During the rewiring phase, gradients and updates are restricted to A_t. The inherited support S_t continues training the current sparse representation, while G_t gives high-gradient off-support coordinates, identified as the primary driver of mask rewiring in Section 3.1, an opportunity to grow, and compete for inclusion in the next sparse mask. After the rewiring phase, magnitude pruning returns the model to the target sparsity, producing the sparse support for the following sparse phase. ARC is therefore a sparse-to-sparse method after the initial dense warmup. It keeps the alternating structure of AC/DC, but compresses the dense rewiring step onto $A_t = S_t \cup G_t$. Larger G_t more closely approximates full dense rewiring, while smaller G_t yields lower theoretical FLOPs.

Combined ARC setting. The three modifications can be applied together: our combined setting uses Zero init, shortened rewiring windows, and sparse updates on $A_t = S_t \cup G_t$. Figure 2b shows that ARC retains the concentration of late sign corrections in rewiring phases. Appendix G gives the full comparison with AC/DC and the differences between the initialization variants.

5 End-to-End Performance and FLOP Tradeoffs

The preceding analysis motivates how to compress dense phases; we now test whether these choices preserve their benefits over complete training runs. We first examine the individual modifications and their accuracy–FLOP tradeoffs on CIFAR100, then evaluate their transfer to ImageNet and language modeling. We consider ResNet18 [26] on CIFAR100 [37], ResNet50 and DeiT-Small [69] on ImageNet [11], and GPT-2 124M [62] on OpenWebText [22]. We compare against AC/DC [60] and RigL [13], and include published Top-KAST [32] results for ResNet50. Our vision runs use three seeds; language-model results use one seed. We report theoretical FLOPs, including support-selection probes. Our implementation uses dense tensors with masks and does not demonstrate wall-clock acceleration. Training details and FLOP accounting are provided in Appendix D.

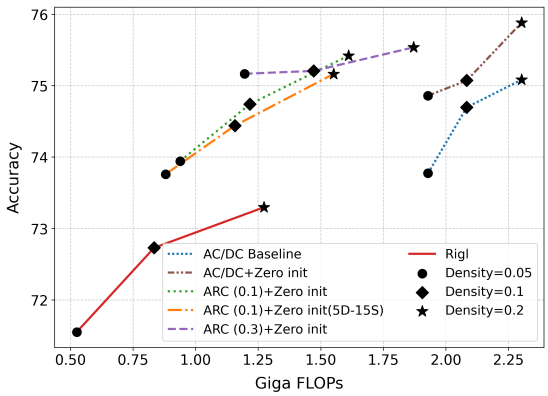


Figure 5: **Performance versus theoretical FLOPs.** Accuracy–FLOP tradeoff for AC/DC, ARC variants, and RigL on CIFAR100. The horizontal axis reports epoch-averaged theoretical training GFLOPs per image, including support-selection probes.

Shorter dense phases preserve performance. We first test whether dense phases must have the same duration as sparse phases. On CIFAR100 with ResNet18, we keep the total cycle length fixed and vary the number of dense epochs per phase. Figure 4 (left) shows that dense phases can be

substantially shortened with little loss in accuracy. In particular, reducing the dense portion from 10 epochs to as few as 3 epochs maintains performance within the observed variation.

This result supports the view that much of the useful rewiring can occur early in the dense phase. Shortening dense phases, therefore, reduces theoretical FLOPs by replacing dense computation with sparse training. The same figure also shows that zero initialization is comparable to or better than stale initialization across dense-phase lengths, suggesting that reintroduced parameters do not need to preserve stale values in order to make effective use of the rewiring window.

ARC matches AC/DC with sparse rewiring. Shortening dense phases reduces how long all coordinates are trained; ARC addresses the complementary question of whether all coordinates need to participate at all. Figure 4 (right) compares ARC with varying G_t densities on CIFAR100. ARC with a sufficiently large rewiring density matches the performance of the AC/DC baseline while reducing theoretical FLOPs, because each rewiring phase updates only $A_t = S_t \cup G_t$ rather than the full dense parameter set. Increasing the density of G_t improves performance: larger rewiring supports better approximate the dense phase, while smaller supports provide greater compression.

Table 1 reports the corresponding CIFAR100 results. ARC(0.1) already performs comparably to the AC/DC baseline across sparsity levels, while ARC(0.3) with zero initialization achieves the best accuracy at 95% and 90% sparsity and is competitive with the best AC/DC variant at 80% sparsity. Combining all three modifications—zero initialization, shorter rewiring windows, and sparse high-gradient rewiring—also remains competitive with AC/DC, despite using substantially fewer theoretical FLOPs.

Table 1: **AC/DC, ARC and RigL on CIFAR100 with ResNet18.**

	Sparsity=0.95	**Sparsity=0.9**	**Sparsity=0.8**
ARC (0.1)	73.91 ± 0.21	74.45 ± 0.14	74.94 ± 0.21
ARC (0.1) + Zero init	73.94 ± 0.26	74.74 ± 0.36	75.42 ± 0.19
ARC (0.3) + Zero init	**75.17** ± 0.98	**75.21** ± 0.26	$\underline{75.54 \pm 0.58}$
ARC (0.1) + Zero init 5-D, 15-S	73.76 ± 0.34	74.44 ± 0.06	75.16 ± 0.20
AC/DC Baseline	73.77 ± 0.17	74.70 ± 0.40	75.08 ± 0.30
AC/DC + Zero init	$\underline{74.86} \pm 0.38$	$\underline{75.07 \pm 0.38}$	**75.88** ± 0.20
RigL	71.55 ± 0.26	72.73 ± 0.41	73.30 ± 0.14

Table 2: **AC/DC, ARC and RigL on ImageNet with DeiT-Small.** We report top-1 accuracy (%) and epoch-averaged theoretical training GFLOPs per image, including support-selection probes.

	Sparsity=0.7		**Sparsity=0.5**	
	Test Acc	**GFLOPs**	**Test Acc**	**GFLOPs**
ARC (0.2)	**76.50** $\pm$ **0.21**	13.34	**77.82** $\pm$ **0.19**	18.11
ARC (0.2) + Zero init	$\underline{76.21 \pm 0.22}$	13.34	77.67 ± 0.76	18.11
ARC (0.2) + Zero init 10-D, 20-S	76.20 ± 0.08	12.59	77.40 ± 0.27	17.36
AC/DC Baseline	75.89 ± 0.31	19.01	77.433 ± 0.30	21.52
AC/DC + Zero init	75.891 ± 0.74	19.01	$\underline{77.82 \pm 0.12}$	21.52
RigL	73.89 ± 0.47	10.32	76.25 ± 0.18	15.30

Comparison with RigL and Pareto frontier. RigL is an important sparse-to-sparse baseline because it also avoids dense training phases through gradient-based rewiring. However, Table 1 shows that all AC/DC-style variants substantially outperform RigL on CIFAR100 at the same sparsity levels. Figure 5 further compares accuracy against theoretical FLOPs. ARC variants improve the performance–FLOP frontier relative to both AC/DC and RigL: they retain AC/DC-like accuracy while shifting to lower FLOP budgets.

ImageNet results. We next evaluate ARC on ImageNet, testing architectural transfer with DeiT-Small and high-sparsity training with ResNet50. For DeiT-Small, Table 2 shows that ARC(0.2)

Table 3: **ResNet50/ImageNet at 95% sparsity.** Published baselines from Table 2 of [60] and our three-seed ARC results. Inference FLOPs per image and full-run training FLOPs are normalized to dense; ARC includes selection probes. Top-KAST uses 95% forward and 50% backward sparsity.

Method	Top-1 (%)	Inference FLOPs	Training FLOPs
Published baselines			
RigL 1× (ERK)	69.70 ± 0.17	$0.12\times$	$0.13\times$
Top-KAST	71.96	$0.08\times$	$0.22\times$
AC/DC	73.14 ± 0.20	$0.11\times$	$0.53\times$
Our runs			
ARC–Stale	72.083 ± 0.077	$0.0920\times$	$0.2321\times$
ARC–Zero	73.087 ± 0.046	$0.1079\times$	$0.2431\times$

Table 4: **GPT-2 124M on OpenWebText at 80% target sparsity.** Results use one training seed (1337) and the end-of-training checkpoint. The zero-shot average is the arithmetic mean of seven task scores (%). Bold indicates the best result among the compared sparse methods.

Method	Validation loss ↓	Perplexity ↓	Zero-shot average ↑
AC/DC	3.13690	23.0327	36.27
RigL	3.13847	23.0685	35.67
ARC–Stale	3.06056	21.3395	36.35
ARC–Zero	**3.05646**	**21.2521**	**37.29**

outperforms standard AC/DC and RigL at both 50% and 70% sparsity, while using fewer theoretical GFLOPs than AC/DC. At 50% sparsity, it also matches AC/DC with zero initialization.

We further evaluate ResNet50 at 95% sparsity using the 100-epoch AC/DC benchmark [60]. Table 3 shows that ARC–Zero nearly matches AC/DC accuracy with 54% fewer theoretical training FLOPs. It also outperforms the published Top-KAST baseline in accuracy at a moderately higher FLOP budget. These results demonstrate that ARC's accuracy–FLOP benefits extend across architectures and to high sparsity. Training details are provided in Appendix D.2, with a hardware-oriented 2:4 semi-structured extension in Appendix E.4.

Language modeling. We also evaluate GPT-2 124M trained from scratch on OpenWebText for 600,000 updates at 80% sparsity. Both ARC variants improve validation perplexity over AC/DC and RigL (Table 4). ARC–Zero reduces AC/DC perplexity from 23.03 to 21.25 (7.7%) and improves six of seven zero-shot task scores, increasing the average from 36.27% to 37.29%. Training details and task-wise results are provided in Appendices D.3 and E.3.

6 Conclusion

We show that the dense phases in AC/DC can be substantially compressed without losing its main benefit. Our theory frames alternating training as sparse consolidation followed by support expansion, and our diagnostics show the corresponding empirical pattern: sparse phases refine the current support, while dense phases expose high-gradient off-support coordinates that drive mask rewiring and persistent sign correction. This suggests that dense phases are not uniformly useful across all parameters. Building on this insight, we introduced Zero init, truncated dense phases, and ARC, which replaces each dense phase with sparse rewiring. Across CIFAR100 and ImageNet, ARC preserves AC/DC-level accuracy while reducing theoretical FLOPs and improving the performance–FLOP frontier relative to AC/DC and RigL. The GPT-2 results extend these benefits to language modeling, with lower perplexity and higher average zero-shot performance. This work focuses on unstructured sparsity, where theoretical FLOP reductions do not always translate directly into wall-clock speedups on current hardware. Nevertheless, unstructured sparsity remains a useful setting for studying the limits of sparse training. Its flexibility allows us to ask which parameters are needed, how sparse masks evolve, and which parts of dense training are essential. The resulting insights can inform both future sparse-training algorithms and more hardware-friendly structured variants.

7 Acknowledgements

We gratefully acknowledge the Gauss Centre for Supercomputing e.V. for funding this project by providing computing time on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC). We are also grateful for funding from the European Research Council (ERC) under the Horizon Europe Framework Programme (HORIZON) for proposal number 101116395 SPARSE-ML.

References

[1] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989.

[2] David Barrett and Benoit Dherin. Implicit gradient regularization. In *International Conference on Learning Representations*, 2021.

[3] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[4] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

[5] Thomas Blumensath and Mike E Davies. Iterative thresholding for sparse approximations. *Journal of Fourier analysis and Applications*, 14(5-6):629–654, 2008.

[6] Rebekka Burkholz. Most activation functions can win the lottery without excessive depth. In *Advances in Neural Information Processing Systems*, 2022.

[7] Rebekka Burkholz, Nilanjana Laha, Rajarshi Mukherjee, and Alkis Gotovos. On the existence of universal lottery tickets. In *International Conference on Learning Representations*, 2022.

[8] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10558–10578, 2024.

[9] 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.

[10] Arthur da Cunha, Emanuele Natale, and Laurent Viennot. Proving the lottery ticket hypothesis for convolutional neural networks. In *International Conference on Learning Representations*, 2022.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[12] Clémentine Carla Juliette Dominé, Nicolas Anguita, Alexandra Maria Proca, Lukas Braun, Daniel Kunin, Pedro A. M. Mediano, and Andrew M Saxe. From lazy to rich: Exact learning dynamics in deep linear networks. In *UniReps: 2nd Edition of the Workshop on Unifying Representations in Neural Models*, 2024.

[13] 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*, pages 2943–2952. PMLR, 2020.

[14] Mathieu Even, Scott Pesme, Suriya Gunasekar, and Nicolas Flammarion. (s)gd over diagonal linear networks: Implicit regularisation, large stepsizes and edge of stability. *ArXiv*, abs/2302.08982, 2023.

[15] Damien Ferbach, Christos Tsirigotis, Gauthier Gidel, and Joey Bose. A general framework for proving the equivariant strong lottery ticket hypothesis. In *International Conference on Learning Representations*, 2023.

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

[17] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Pruning neural networks at initialization: Why are we missing the mark? In *International Conference on Learning Representations*, 2021.

[18] Advait Gadhikar and Rebekka Burkholz. Masks, signs, and learning rate rewinding. In *Twelfth International Conference on Learning Representations*, 2024.

[19] Advait Gadhikar, Tom Jacobs, Chao Zhou, and Rebekka Burkholz. Sign-in to the lottery: Reparameterizing sparse training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[20] Advait Harshal Gadhikar, Sohom Mukherjee, and Rebekka Burkholz. Why random pruning is all we need to start sparse. In *International Conference on Machine Learning*, 2023.

[21] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024.

[22] Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`, 2019.

[23] Suriya Gunasekar, Blake Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nathan Srebro. Implicit regularization in matrix factorization, 2017.

[24] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[25] B. Hassibi, D.G. Stork, and G.J. Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, pages 293–299 vol.1, 1993.

[26] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

[27] Tom Jacobs and Rebekka Burkholz. Mask in the mirror: Implicit sparsification. In *The Thirteenth International Conference on Learning Representations*, 2025.

[28] Tom Jacobs, Advait Gadhikar, Celia Rubio-Madrigal, and Rebekka Burkholz. Ham: A hyperbolic step to regulate implicit bias. *International Conference on Learning Representations*, 2026.

[29] Tom Jacobs, Chao Zhou, and Rebekka Burkholz. Mirror, mirror of the flow: How does regularization shape implicit bias? In *Forty-second International Conference on Machine Learning*, 2025.

[30] Tom Jacobs, Chao Zhou, and Rebekka Burkholz. Never saddle: Reparameterized steepest descent as mirror flow. In *The Fourteenth International Conference on Learning Representations*, 2026.

[31] Arthur Jacot, François Ged, Berfin Şimşek, Clément Hongler, and Franck Gabriel. Saddle-to-saddle dynamics in deep linear networks: Small initialization training, symmetry, and sparsity. *arXiv preprint arXiv:2106.15933*, 2021.

[32] Siddhant Jayakumar, Razvan Pascanu, Jack Rae, Simon Osindero, and Erich Elsen. Top-kast: Top-k always sparse training. *Advances in Neural Information Processing Systems*, 33:20744–20754, 2020.

[33] Andrej Karpathy. nanogpt, 2022.

[34] Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, volume 29, pages 586–594, 2016.

[35] Chris Kolb, Laetitia Frost, Bernd Bischl, and David Rügamer. Differentiable sparsity via d-gating: Simple and versatile structured penalization. *ArXiv*, abs/2509.23898, 2025.

[36] Chris Kolb, Tobias Weber, Bernd Bischl, and David Rügamer. Deep weight factorization: Sparse learning through the lens of artificial symmetries. In *The Thirteenth International Conference on Learning Representations*, 2025.

[37] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[38] Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *Proceedings of the International Conference on Machine Learning*, July 2020.

[39] Denis Kuznedelev, Eldar Kurtić, Elias Frantar, and Dan Alistarh. Cap: Correlation-aware pruning for highly-accurate sparse vision models. *Advances in Neural Information Processing Systems*, 36, 2024.

[40] Mike Lasby, Anna Golubeva, Utku Evci, Mihai Nica, and Yani Ioannou. Dynamic sparse training with structured sparsity. *arXiv preprint arXiv:2305.02299*, 2023.

[41] Thomas Laurent and James von Brecht. Deep linear networks with arbitrary loss: All local minima are global. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 2902–2907, 2018.

[42] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2019.

[43] Zhiyuan Li, Tianhao Wang, Jason D. Lee, and Sanjeev Arora. Implicit bias of gradient descent on reparametrized models: On equivalence to mirror descent. *ArXiv*, abs/2207.04036, 2022.

[44] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Li Shen, Decebal Constantin Mocanu, Zhangyang Wang, and Mykola Pechenizkiy. The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. In *International Conference on Learning Representations*, 2021.

[45] 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*, pages 6989–7000. PMLR, 2021.

[46] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018.

[47] Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, 2020.

[48] 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*, pages 2381–2391, 2018.

[49] 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), jun 2018.

[50] Sree Harsha Nelaturu, Advait Gadhikar, and Rebekka Burkholz. TurboPrune: High-Speed Distributed Lottery Ticket Training, 2024.

[51] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017.

[52] NVIDIA. TensorRT documentation: Data formats and tensors. `https://docs.nvidia.com/deeplearning/tensorrt/latest/inference-library/data-formats-tensors.html`. Section on structured sparsity.

[53] Laurent Orseau, Marcus Hutter, and Omar Rivasplata. Logarithmic pruning is all you need. *Advances in Neural Information Processing Systems*, 33, 2020.

[54] 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 (volume 1: Long papers)*, pages 1525–1534, 2016.

[55] Shreyas Malakarjun Patil and Constantine Dovrolis. Phew: Constructing sparse networks that learn fast and generalize well without training data. In *International Conference on Machine Learning*, pages 8432–8442. PMLR, 2021.

[56] Mansheej Paul, Feng Chen, Brett W. Larsen, Jonathan Frankle, Surya Ganguli, and Gintare Karolina Dziugaite. Unmasking the lottery ticket hypothesis: What's encoded in a winning ticket's mask? In *The Eleventh International Conference on Learning Representations*, 2023.

[57] Ankit Pensia, Shashank Rajput, Alliot Nagle, Harit Vishwakarma, and Dimitris Papailiopoulos. Optimal lottery tickets via subset sum: Logarithmic over-parameterization is sufficient. In *Advances in Neural Information Processing Systems*, volume 33, pages 2599–2610, 2020.

[58] Scott Pesme and Nicolas Flammarion. Saddle-to-saddle dynamics in diagonal linear networks, 2023.

[59] Scott Pesme, Loucas Pillaud-Vivien, and Nicolas Flammarion. Implicit bias of sgd for diagonal linear networks: a provable benefit of stochasticity, 2021.

[60] Alexandra Peste, Eugenia Iofinova, Adrian Vladu, and Dan Alistarh. Ac/dc: Alternating compressed/decompressed training of deep neural networks, 2021.

[61] Hoang Pham, The-Anh Ta, Shiwei Liu, Lichuan Xiang, Dung D. Le, Hongkai Wen, and Long Tran-Thanh. Towards data-agnostic pruning at initialization: What makes a good sparse mask? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[62] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[63] Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations*, 2020.

[64] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[65] Karthik Abinav Sankararaman, Soham De, Zheng Xu, W. Ronny Huang, and Tom Goldstein. The impact of neural network overparameterization on gradient confusion and stochastic gradient descent. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8469–8479. PMLR, 13–18 Jul 2020.

[66] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations*, 2014.

[67] Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. *Advances in Neural Information Processing Systems*, 33:18098–18109, 2020.

[68] Hidenori Tanaka, Daniel Kunin, Daniel L. Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Advances in Neural Information Processing Systems*, 2020.

[69] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[70] Tomas Vaškevičius, Varun Kanade, and Patrick Rebeschini. Implicit regularization for optimal sparse recovery, 2019.

[71] Chaoqi Wang, Guodong Zhang, and Roger B. Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020.

[72] Shuyang Wang and Diego Klabjan. A mirror descent perspective of smoothed sign descent. In *Conference on Uncertainty in Artificial Intelligence*, 2024.

[73] Blake Woodworth, Suriya Gunasekar, Jason D. Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models, 2020.

[74] Xin Yu, Thiago Serra, Srikumar Ramalingam, and Shandian Zhe. The combinatorial brain surgeon: Pruning weights that cancel one another in neural networks, 2022.

[75] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

[76] 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*, pages 4791–4800, 2019.

[77] Yedi Zhang, Andrew M. Saxe, and Peter E. Latham. Saddle-to-saddle dynamics explains a simplicity bias across neural network architectures. In *The Fourteenth International Conference on Learning Representations*, 2026.

[78] Yingtao Zhang, Jialin Zhao, Wenjing Wu, Ziheng Liao, Umberto Michieli, and Carlo Vittorio Cannistraci. Brain-inspired sparse training enables transformers and llms to perform as fully connected. *arXiv preprint arXiv:2501.19107*, 2025.

[79] Chao Zhou, Tom Jacobs, Advait Gadhikar, and Rebekka Burkholz. Pay attention to small weights. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[80] Liu Ziyin and Zihao Wang. spred: Solving l_1 penalty with sgd, 2023.

A AC/DC and ARC Schematic

Figure 6 shows the schematic for both AC/DC and ARC.

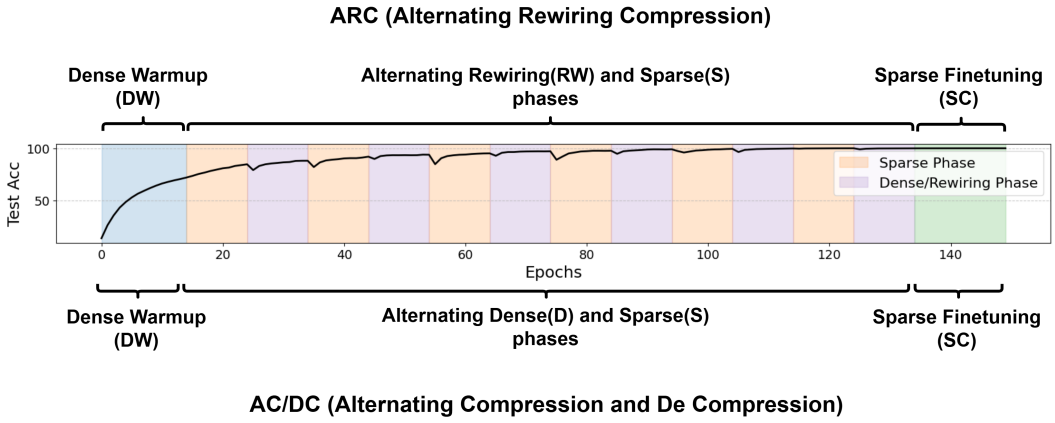


Figure 6: **AC/DC & ARC Schematic.** AC/DC finds a sparse network, by starting with a dense warmup, then alternating between dense and sparse phases and finally a sparse finetuning phase. ARC replaces the alternating dense phases with sparse rewiring phases.

B Convergence theory

The main goal of this section is to prove convergence of a modified iterative hard thresholding algorithm that replaces dense training phases with sparse ones that compress the original dense gradient. Our theory relies on the following assumptions.

Assumption B.1 (Sparse optimum)**.** *There exists a global minimizer* $\theta^\star \in \arg\min_\theta f(\theta)$ *such that*

$$\|\theta^\star\|_0 \le k^\star.$$

Assumption B.2 (Restricted smoothness)**.** *The objective* $f : \mathbb{R}^N \to \mathbb{R}$ *is* (s, β)*-smooth with*

$$s = 2k + q + 3k^\star.$$

That is, for all θ *and all* s*-sparse perturbations* δ*,*

$$f(\theta + \delta) \le f(\theta) + \langle \nabla f(\theta), \delta \rangle + \frac{\beta}{2}\|\delta\|_2^2.$$

Assumption B.3 (Concentrated PL condition)**.** *There exists* $\alpha > 0$ *such that* f *satisfies the* $(k^\star, \alpha)$*-concentrated PL condition defined in* (1)*.*

Assumption B.4 (Compressed-gradient accuracy)**.** *Let*

$$\Delta_t = f(\theta_t) - f(\theta^\star).$$

For every support U_t *arising in the IHT proof, with*

$$|U_t| \le 2k + q + 3k^\star,$$

the compressed gradient $h_t = P_{A_t} g_t$ *satisfies*

$$\mathbb{E}\big[\|(h_t - \nabla f(\theta_t))_{U_t}\|_2^2 \,\big|\, \theta_t\big] \le \sigma_{\mathrm{cg}}^2 + \rho\,\Delta_t.$$

Here σ_{cg}^2 *is the stochastic-gradient variance remaining after projection, and* $\rho\Delta_t$ *is the deterministic error introduced by gradient compression.*

Lemma B.5 (One-step AC/DC bound with compressed gradients)**.** *Suppose Assumptions B.1, B.2, and B.3 hold. Let*

$$\theta_{t+1} = T_k(\theta_t - \eta h_t), \qquad \eta = \frac{1}{2\beta}.$$

There exist universal constants $C_0, C_1 > 0$ such that, whenever

$$k \geq C_0 k^\star \left(\frac{\beta}{\alpha}\right)^2,$$

we have

$$\mathbb{E}[\Delta_{t+1} \mid \theta_t] \leq \left(1 - \frac{\alpha}{32\beta}\right) \Delta_t + C_1 \beta \, \mathbb{E}\left[\|(h_t - \nabla f(\theta_t))_{U_t}\|_2^2 \,\middle|\, \theta_t\right].$$

Proof. Let

$$S = \operatorname{supp}(\theta_t), \qquad S_0 = \operatorname{supp}(\theta_{t+1}), \qquad S^\star = \operatorname{supp}(\theta^\star),$$

and let U_t denote the union of the supports appearing in the IHT projection argument:

$$U_t \supseteq S \cup S_0 \cup S^\star \cup A_t, \qquad |U_t| \leq 2k + q + 3k^\star.$$

By restricted smoothness,

$$f(\theta_{t+1}) \leq f(\theta_t) + \langle \nabla f(\theta_t), \theta_{t+1} - \theta_t \rangle + \frac{\beta}{2} \|\theta_{t+1} - \theta_t\|_2^2.$$

As in the stochastic-IHT proof for AC/DC, insert and subtract the ideal dense-gradient step:

$$\theta_t - \frac{1}{\beta} \nabla f(\theta_t).$$

This gives

$$f(\theta_{t+1}) \leq f(\theta_t) + \frac{\beta}{2} \left\| \theta_{t+1} - \left(\theta_t - \frac{1}{\beta} \nabla f(\theta_t)\right) \right\|_2^2 - \frac{1}{2\beta} \|\nabla f(\theta_t)\|_{U_t}^2.$$

Now write

$$\theta_{t+1} = T_k(\theta_t - \eta h_t).$$

The standard AC/DC/IHT projection argument controls the error introduced by applying T_k, provided

$$k \gtrsim k^\star (\beta/\alpha)^2.$$

In the exact-gradient case $h_t = \nabla f(\theta_t)$, this yields

$$\Delta_{t+1} \leq \left(1 - \frac{\alpha}{32\beta}\right) \Delta_t.$$

When $h_t \neq \nabla f(\theta_t)$, every occurrence of the stochastic gradient in the AC/DC proof is replaced by

$$h_t = \nabla f(\theta_t) + e_t, \qquad e_t = h_t - \nabla f(\theta_t).$$

The additional terms are supported on U_t and are bounded by Young's inequality:

$$2|\langle a, e_t \rangle| \leq \lambda \|a\|_2^2 + \lambda^{-1} \|e_t\|_2^2.$$

Choosing λ as in the standard IHT proof, the terms involving $\|a\|_2^2$ are absorbed into the dense-gradient contraction, while the remaining contribution is at most

$$C_1 \beta \|(h_t - \nabla f(\theta_t))_{U_t}\|_2^2.$$

Taking conditional expectation gives the claim. □

Theorem B.6 (AC/DC with compressed gradient)**.** *Suppose Assumptions B.1–B.4 hold and*

$$k \geq C_0 k^\star \left(\frac{\beta}{\alpha}\right)^2.$$

Define

$$\lambda_{\mathrm{cg}} = \frac{\alpha}{32\beta} - C_1 \beta \rho.$$

If

$$\lambda_{\mathrm{cg}} > 0,$$

then the compressed-gradient AC/DC update satisfies

$$\mathbb{E}[\Delta_{t+1}] \leq (1 - \lambda_{\mathrm{cg}})\mathbb{E}[\Delta_t] + C_1\beta\sigma_{\mathrm{cg}}^2.$$

Consequently, after T iterations,

$$\mathbb{E}[\Delta_T] \leq (1 - \lambda_{\mathrm{cg}})^T\Delta_0 + \frac{C_1\beta\sigma_{\mathrm{cg}}^2}{\lambda_{\mathrm{cg}}}.$$

Equivalently, for any tolerance $\varepsilon > 0$, after

$$T = O\left(\frac{1}{\lambda_{\mathrm{cg}}}\log\frac{\Delta_0}{\varepsilon}\right)$$

iterations, the method reaches

$$\mathbb{E}[f(\theta_T) - f(\theta^\star)] \leq \varepsilon + O\left(\frac{\beta\sigma_{\mathrm{cg}}^2}{\lambda_{\mathrm{cg}}}\right).$$

Proof. By Lemma B.5,

$$\mathbb{E}[\Delta_{t+1} \mid \theta_t] \leq \left(1 - \frac{\alpha}{32\beta}\right)\Delta_t + C_1\beta\,\mathbb{E}\left[\|(h_t - \nabla f(\theta_t))_{U_t}\|_2^2 \,\middle|\, \theta_t\right].$$

Using Assumption B.4,

$$\mathbb{E}[\Delta_{t+1} \mid \theta_t] \leq \left(1 - \frac{\alpha}{32\beta} + C_1\beta\rho\right)\Delta_t + C_1\beta\sigma_{\mathrm{cg}}^2.$$

By definition,

$$\lambda_{\mathrm{cg}} = \frac{\alpha}{32\beta} - C_1\beta\rho,$$

so

$$\mathbb{E}[\Delta_{t+1} \mid \theta_t] \leq (1 - \lambda_{\mathrm{cg}})\Delta_t + C_1\beta\sigma_{\mathrm{cg}}^2.$$

Taking total expectation and unrolling the recursion gives

$$\mathbb{E}[\Delta_T] \leq (1 - \lambda_{\mathrm{cg}})^T\Delta_0 + C_1\beta\sigma_{\mathrm{cg}}^2\sum_{j=0}^{T-1}(1 - \lambda_{\mathrm{cg}})^j.$$

Since

$$\sum_{j=0}^{T-1}(1 - \lambda_{\mathrm{cg}})^j \leq \frac{1}{\lambda_{\mathrm{cg}}},$$

we obtain

$$\mathbb{E}[\Delta_T] \leq (1 - \lambda_{\mathrm{cg}})^T\Delta_0 + \frac{C_1\beta\sigma_{\mathrm{cg}}^2}{\lambda_{\mathrm{cg}}}.$$

The stated iteration complexity follows by choosing T so that

$$(1 - \lambda_{\mathrm{cg}})^T\Delta_0 \leq \varepsilon.$$

□

B.1 Verifying the compressed-gradient assumption

The previous theorem reduces the analysis to proving that

$$P_{A_t}g_t$$

is a sufficiently accurate proxy for the dense gradient on the supports relevant to the IHT proof. The next proposition shows that this follows from gradient compressibility outside the inherited sparse mask.

Assumption B.7 (Old support plus growth support captures gradient energy)**.** *For every iterate θ_t, let $S_t = \mathrm{supp}(\theta_t)$. Suppose that there exists a set $B_t \subseteq S_t^c$, with $|B_t| = q$, such that*

$$\|(I - P_{S_t \cup B_t})\nabla f(\theta_t)\|_{U_t}^2 \leq \tau + \rho\Delta_t.$$

Proposition B.8 (Top-gradient growth gives the best proxy support)**.** *Let*

$$G_t = \mathrm{Top}_q\big((\nabla f(\theta_t))_{S_t^c}\big), \qquad A_t = S_t \cup G_t.$$

If Assumption B.7 holds, then

$$\|(I - P_{A_t})\nabla f(\theta_t)\|_{U_t}^2 \leq \tau + \rho\Delta_t.$$

Therefore, top-gradient growth captures at least as much dense-gradient energy as any other q-coordinate growth set outside the inherited support.

Proof. The operator Top_q gives the best q-sparse approximation in Euclidean norm. Therefore, among all sets $B \subseteq S_t^c$ with $|B| = q$,

$$\|(I - P_{S_t \cup G_t})\nabla f(\theta_t)\|_{U_t}^2 \leq \|(I - P_{S_t \cup B})\nabla f(\theta_t)\|_{U_t}^2.$$

Taking $B = B_t$ from Assumption B.7 proves the result. □

Proposition B.9 (From support capture to compressed-gradient accuracy)**.** *Suppose g_t is a stochastic gradient satisfying*

$$\mathbb{E}[g_t \mid \theta_t] = \nabla f(\theta_t), \qquad \mathbb{E}[\|(g_t - \nabla f(\theta_t))_{A_t}\|_2^2 \mid \theta_t] \leq \sigma_A^2.$$

Assume also that

$$\|(I - P_{A_t})\nabla f(\theta_t)\|_{U_t}^2 \leq \tau + \rho\Delta_t.$$

Then $h_t = P_{A_t} g_t$ satisfies Assumption B.4 with

$$\sigma_{\mathrm{cg}}^2 = 2\sigma_A^2 + 2\tau, \qquad \rho_{\mathrm{cg}} = 2\rho.$$

Proof. We decompose

$$h_t - \nabla f(\theta_t) = P_{A_t}(g_t - \nabla f(\theta_t)) - (I - P_{A_t})\nabla f(\theta_t).$$

Restricting to U_t and using $\|a + b\|_2^2 \leq 2\|a\|_2^2 + 2\|b\|_2^2$,

$$\|(h_t - \nabla f(\theta_t))_{U_t}\|_2^2 \leq 2\|P_{A_t}(g_t - \nabla f(\theta_t))\|_2^2 + 2\|(I - P_{A_t})\nabla f(\theta_t)\|_{U_t}^2.$$

Taking conditional expectation gives

$$\mathbb{E}[\|(h_t - \nabla f(\theta_t))_{U_t}\|_2^2 \mid \theta_t] \leq 2\sigma_A^2 + 2\tau + 2\rho\Delta_t.$$

Thus Assumption B.4 holds with

$$\sigma_{\mathrm{cg}}^2 = 2\sigma_A^2 + 2\tau, \qquad \rho_{\mathrm{cg}} = 2\rho.$$

□

C Saddle-to-saddle learning dynamics explain benefits of alternating phases

The convergence theory leaves us to understand the role of the alternating sparse phases during training. Our main intuitive argument is that saddle-to-saddle dynamics that learn singular directions in order of the dominant singular values focus learning on a low dimensional subspace. As long as the dynamics can be compressed to a sparse support, we naturally have to change the support when we enter a new learning phase to expand our low dimensional subspace by a new singular direction.

Again, we consider a three-layer linear network with one hidden layer,

$$f_{W_1,W_2}(x) = W_2 W_1 x,$$

trained on a population squared loss with whitened inputs. Let

$$M = \mathbb{E}[yx^\top]$$

denote the target input-output map, and write its singular value decomposition as

$$M = U \operatorname{diag}(s_1, \dots, s_r) V^\top, \qquad s_1 > s_2 > \cdots > s_r > 0.$$

The population loss is

$$\mathcal{L}(W_1, W_2) = \frac{1}{2} \|W_2 W_1 - M\|_F^2.$$

We write

$$\theta = (\operatorname{vec}(W_1), \operatorname{vec}(W_2)), \qquad g(\theta) = \nabla \mathcal{L}(\theta).$$

Similarly to previous section, for a coordinate set A, let P_A denote the coordinate projection that keeps the coordinates in A and sets all other coordinates to zero. At the start of a sparse-growth phase, let S be the support inherited from the previous magnitude-pruned phase. The growth phase activates

$$A = S \cup G,$$

where

$$G = \operatorname{Top}_q(g(\theta_0)_{S^c})$$

is the set of the q largest-magnitude gradient coordinates outside S, computed at the beginning of the phase. During the phase we train only coordinates in A:

$$\dot{\theta}_A(t) = -P_A g(\theta_A(t)).$$

The dense gradient flow is

$$\dot{\theta}_\star(t) = -g(\theta_\star(t)).$$

Assumption C.1 (Gradient capture by the inherited-plus-growth support in rewiring phase)**.** *Along the rewiring phase, the active support $A = S \cup G$ captures all but an ε-fraction of the dense gradient energy:*

$$\|(I - P_A) g(\theta_A(t))\|_2 \le \varepsilon \|g(\theta_A(t))\|_2 \qquad \textit{for all } t \in [0, T].$$

Theorem C.2 (Sparse-growth phases are dense-gradient perturbations)**.** *Assume $\mathcal{L}$ is L-smooth on a tube containing the dense and sparse-growth trajectories. Suppose Assumption C.1 holds and that $\|g(\theta_A(t))\|_2 \le G$ for $t \in [0, T]$. If $\theta_A(0) = \theta_\star(0)$, then*

$$\|\theta_A(t) - \theta_\star(t)\|_2 \le \frac{\varepsilon G}{L} \left(e^{Lt} - 1\right) \qquad \textit{for all } t \in [0, T].$$

Moreover, the instantaneous loss decrease during the sparse-growth phase satisfies

$$\frac{d}{dt} \mathcal{L}(\theta_A(t)) = -\|P_A g(\theta_A(t))\|_2^2 \le -(1 - \varepsilon^2) \|g(\theta_A(t))\|_2^2.$$

Thus the sparse-growth phase achieves at least a $1 - \varepsilon^2$ fraction of the dense gradient-flow decrease in loss.

Proof. Write the sparse-growth dynamics as

$$\dot{\theta}_A = -P_A g(\theta_A) = -g(\theta_A) + r_A(\theta_A),$$

where

$$r_A(\theta_A) = (I - P_A) g(\theta_A).$$

By Assumption C.1,

$$\|r_A(\theta_A(t))\|_2 \le \varepsilon \|g(\theta_A(t))\|_2 \le \varepsilon G.$$

Let $e(t) = \theta_A(t) - \theta_\star(t)$. Then

$$\dot{e}(t) = -g(\theta_A(t)) + g(\theta_\star(t)) + r_A(\theta_A(t)).$$

Using L-Lipschitzness of g,

$$\frac{d}{dt} \|e(t)\|_2 \le L \|e(t)\|_2 + \varepsilon G.$$

Since $e(0) = 0$, Gronwall's inequality gives

$$\|e(t)\|_2 \le \frac{\varepsilon G}{L} \left(e^{Lt} - 1\right).$$

For the loss decrease,

$$\frac{d}{dt}\mathcal{L}(\theta_A(t)) = \langle g(\theta_A(t)), \dot{\theta}_A(t)\rangle = -\langle g(\theta_A(t)), P_A g(\theta_A(t))\rangle.$$

Since P_A is an orthogonal coordinate projection,

$$\langle g, P_A g\rangle = \|P_A g\|_2^2.$$

Furthermore,

$$\|P_A g\|_2^2 = \|g\|_2^2 - \|(I - P_A)g\|_2^2 \geq (1 - \varepsilon^2)\|g\|_2^2,$$

which proves the claim. □

C.1 Preservation of SVD-mode learning order

Following the dense deep-linear analysis of Saxe et al., suppose the dense trajectory is initialized on the SVD-aligned manifold

$$W_2 = U\,\mathrm{diag}(b_i)R^\top, \qquad W_1 = R\,\mathrm{diag}(a_i)V^\top.$$

Then the dense dynamics decouple into singular modes:

$$\dot{a}_i = b_i(s_i - a_i b_i), \qquad \dot{b}_i = a_i(s_i - a_i b_i).$$

Let

$$m_i(\theta) = a_i(\theta)b_i(\theta)$$

denote the learned strength of singular mode i.

Assumption C.3 (Mode-separation margin)**.** *On a time interval $I \subseteq [0, T]$, the dense dynamics have a strict mode velocity gap:*

$$\dot{m}_i^\star(t) - \dot{m}_j^\star(t) \geq \gamma \qquad \textit{for all } t \in I,$$

for some $\gamma > 0$, where i is the dominant mode and j is a competing lower singular mode.

Theorem C.4 (Sparse growth preserves mode ordering)**.** *Assume Assumptions C.1 and C.3. Suppose the mode coordinates m_i and m_j are differentiable in a neighborhood of the trajectories and satisfy*

$$\|\nabla m_i(\theta)\|_2 \leq B, \qquad \|\nabla m_j(\theta)\|_2 \leq B.$$

If

$$2B\varepsilon G < \gamma,$$

then the sparse-growth phase preserves the dense ordering of the two mode velocities:

$$\dot{m}_i^A(t) > \dot{m}_j^A(t) \qquad \textit{for all } t \in I.$$

In particular, whenever the dense dynamics learn mode i before mode j with a margin γ, the sparse-growth phase preserves this ordering up to perturbation error controlled by the discarded gradient tail.

Proof. Along dense flow,

$$\dot{m}_\ell^\star(t) = -\langle \nabla m_\ell(\theta_\star(t)), g(\theta_\star(t))\rangle.$$

Along sparse-growth flow,

$$\dot{m}_\ell^A(t) = -\langle \nabla m_\ell(\theta_A(t)), P_A g(\theta_A(t))\rangle.$$

Ignoring the trajectory-displacement error, which is controlled by Theorem C.2, the direct vector-field perturbation is

$$|\langle \nabla m_\ell(\theta), g(\theta)\rangle - \langle \nabla m_\ell(\theta), P_A g(\theta)\rangle| = |\langle \nabla m_\ell(\theta), (I - P_A)g(\theta)\rangle|\,.$$

By Cauchy–Schwarz and Assumption C.1,

$$|\langle \nabla m_\ell(\theta), (I - P_A)g(\theta)\rangle| \leq B\varepsilon G.$$

Therefore each mode velocity changes by at most $B\varepsilon G$. Hence the pairwise gap changes by at most $2B\varepsilon G$. Since the dense gap is at least γ, we have

$$\dot{m}_i^A(t) - \dot{m}_j^A(t) \geq \gamma - 2B\varepsilon G > 0.$$

Thus the sparse-growth phase preserves the dense ordering. □

C.2 Why sparse initialization can induce gradient compressibility

The previous results reduce the problem to verifying Assumption C.1. We now give a sufficient condition under which the dense gradient is entrywise compressible.

Let

$$E = M - W_2 W_1$$

denote the residual target map. The gradients are

$$\nabla_{W_2}\mathcal{L} = -EW_1^\top, \qquad \nabla_{W_1}\mathcal{L} = -W_2^\top E.$$

Suppose the residual is approximately low rank in the target singular basis:

$$E = \sum_{\ell=1}^{r} \rho_\ell u_\ell v_\ell^\top + \Delta.$$

Define the hidden responses

$$h_\ell = W_1 v_\ell, \qquad z_\ell = W_2^\top u_\ell.$$

Then

$$\nabla_{W_2}\mathcal{L} = -\sum_{\ell=1}^{r} \rho_\ell u_\ell h_\ell^\top - \Delta W_1^\top,$$

and

$$\nabla_{W_1}\mathcal{L} = -\sum_{\ell=1}^{r} \rho_\ell z_\ell v_\ell^\top - W_2^\top \Delta.$$

Thus the gradients are sums of outer products involving the hidden responses h_ℓ and z_ℓ.

Lemma C.5 (Outer-product compressibility)**.** *Let $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^n$. Suppose there exist coordinate sets $I \subseteq [m]$ and $J \subseteq [n]$ such that*

$$\|x_{I^c}\|_2 \le \eta_x \|x\|_2, \qquad \|y_{J^c}\|_2 \le \eta_y \|y\|_2.$$

Then the support $I \times J$ captures at least

$$(1-\eta_x^2)(1-\eta_y^2)$$

of the Frobenius energy of $xy^\top$. Equivalently,

$$\|(I - P_{I\times J})(xy^\top)\|_F \le \sqrt{\eta_x^2 + \eta_y^2 - \eta_x^2\eta_y^2}\, \|xy^\top\|_F.$$

Since entrywise top-k gives the best k-sparse approximation in Frobenius norm, the same upper bound holds for $T_k(xy^\top)$ with $k = |I||J|$.

Proof. We have

$$\|P_{I\times J}(xy^\top)\|_F^2 = \|x_I\|_2^2 \|y_J\|_2^2.$$

By assumption,

$$\|x_I\|_2^2 \ge (1-\eta_x^2)\|x\|_2^2, \qquad \|y_J\|_2^2 \ge (1-\eta_y^2)\|y\|_2^2.$$

Therefore

$$\|P_{I\times J}(xy^\top)\|_F^2 \ge (1-\eta_x^2)(1-\eta_y^2)\|x\|_2^2\|y\|_2^2.$$

Since

$$\|xy^\top\|_F^2 = \|x\|_2^2\|y\|_2^2,$$

the residual satisfies

$$\|(I - P_{I\times J})(xy^\top)\|_F^2 \le \big(1 - (1-\eta_x^2)(1-\eta_y^2)\big)\, \|xy^\top\|_F^2.$$

This gives the stated bound. The final claim follows because entrywise top-k is the minimizer of the Frobenius error over all k-sparse matrices. □

Assumption C.6 (Sparse-response condition induced by sparse initialization)**.** *For the residual modes that dominate training during the current phase, the hidden responses $h_\ell = W_1 v_\ell$ and $z_\ell = W_2^\top u_\ell$ are entrywise compressible. That is, for each ℓ, there exist supports H_ℓ, Z_ℓ with*

$$|H_\ell| = p_{h,\ell}, \qquad |Z_\ell| = p_{z,\ell},$$

such that

$$\|(h_\ell)_{H_\ell^c}\|_2 \le \eta_{h,\ell}\|h_\ell\|_2, \qquad \|(z_\ell)_{Z_\ell^c}\|_2 \le \eta_{z,\ell}\|z_\ell\|_2.$$

Proposition C.7 (Sparse responses imply compressible dense gradients)**.** *Assume the residual decomposition above and Assumption C.6. Let*

$$K_2 = d_{\text{out}} \sum_{\ell=1}^{r} p_{h,\ell}, \qquad K_1 = d_{\text{in}} \sum_{\ell=1}^{r} p_{z,\ell}.$$

Then

$$\|\nabla_{W_2}\mathcal{L} - T_{K_2}(\nabla_{W_2}\mathcal{L})\|_F \le \sum_{\ell=1}^{r} |\rho_\ell|\, \eta_{h,\ell}\|h_\ell\|_2 + \|\Delta\|_F \|W_1\|_{\text{op}},$$

and

$$\|\nabla_{W_1}\mathcal{L} - T_{K_1}(\nabla_{W_1}\mathcal{L})\|_F \le \sum_{\ell=1}^{r} |\rho_\ell|\, \eta_{z,\ell}\|z_\ell\|_2 + \|W_2\|_{\text{op}} \|\Delta\|_F.$$

Consequently, if the residual is dominated by a small number of modes and the sparse initialization induces compressible hidden responses, then the dense gradient is entrywise compressible.

Proof. Consider the W_2 gradient:

$$\nabla_{W_2}\mathcal{L} = -\sum_{\ell=1}^{r} \rho_\ell u_\ell h_\ell^\top - \Delta W_1^\top.$$

For each term $u_\ell h_\ell^\top$, take the full support of u_ℓ and the compressible support H_ℓ of h_ℓ. Applying Lemma C.5 with $\eta_x = 0$ and $\eta_y = \eta_{h,\ell}$ gives

$$\left\|u_\ell h_\ell^\top - P_{[d_{\text{out}}]\times H_\ell}(u_\ell h_\ell^\top)\right\|_F \le \eta_{h,\ell}\|u_\ell h_\ell^\top\|_F.$$

Since $\|u_\ell\|_2 = 1$, this becomes

$$\left\|u_\ell h_\ell^\top - P_{[d_{\text{out}}]\times H_\ell}(u_\ell h_\ell^\top)\right\|_F \le \eta_{h,\ell}\|h_\ell\|_2.$$

Taking the union of these supports over ℓ gives a support of size at most K_2. By the triangle inequality,

$$\|\nabla_{W_2}\mathcal{L} - P_{\Gamma_2}\nabla_{W_2}\mathcal{L}\|_F \le \sum_{\ell=1}^{r} |\rho_\ell|\eta_{h,\ell}\|h_\ell\|_2 + \|\Delta W_1^\top\|_F.$$

Finally,

$$\|\Delta W_1^\top\|_F \le \|\Delta\|_F \|W_1\|_{\text{op}}.$$

Since T_{K_2} is the best K_2-sparse approximation, it does no worse than the particular support Γ_2, proving the first bound.

The proof for $\nabla_{W_1}\mathcal{L}$ is identical, using

$$\nabla_{W_1}\mathcal{L} = -\sum_{\ell=1}^{r} \rho_\ell z_\ell v_\ell^\top - W_2^\top \Delta,$$

the full support of v_ℓ, and the compressible support Z_ℓ of z_ℓ. □

Corollary C.8 (Sufficient condition for rewiring phase)**.** *Suppose the inherited support S contains the large-magnitude coordinates produced by previously learned modes, and the added support G contains the largest coordinates of the gradient outside S. If Proposition C.7 implies*

$$\|(I - P_{S\cup G})g(\theta)\|_2 \le \varepsilon\|g(\theta)\|_2$$

with ε small enough that

$$2B\varepsilon G < \gamma,$$

then the rewiring phase preserves the dominant-singular-mode ordering of the dense saddle-to-saddle dynamics on the interval where the dense mode gap is at least γ.

Summary. The main intuitive message is: Sparse growth preserves the saddle-to-saddle dynamics when $S \cup G$ captures most dense-gradient energy.

Sparse initialization alone does not guarantee entrywise gradient compressibility for arbitrary singular vectors. The sparse-response condition provides a sufficient condition for such compressibility, consistent with the motivation behind our empirical diagnostics.

C.3 Alternating sparse phases as sparse saddle-to-saddle dynamics

We now interpret alternating sparse phases through the lens of saddle-to-saddle learning dynamics. Dense gradient flow in deep linear networks learns the target map by successively increasing the effective rank of the end-to-end predictor. We argue that alternating sparse phases implement the same mechanism in a sparse parameter space: one phase consolidates the current low-rank sparse solution, while the next phase rewires the support to activate coordinates needed for the next singular mode.

Let

$$\mathcal{L}(W_1, W_2) = \frac{1}{2}\|W_2 W_1 - M\|_F^2, \qquad M = U \operatorname{diag}(s_1, \ldots, s_r) V^\top,$$

where

$$s_1 > s_2 > \cdots > s_r > 0.$$

Let $\theta = (W_1, W_2)$, and let $g(\theta) = \nabla \mathcal{L}(\theta)$. For a support S, write P_S for coordinate projection onto S, and define the support-restricted flow

$$\dot{\theta} = -P_S g(\theta).$$

We consider an alternating sparse algorithm indexed by phases $m = 0, 1, \ldots, r-1$. At phase m, the algorithm has a sparse support S_m. It first trains on S_m, then constructs a growth support

$$G_m = \operatorname{Top}_q\big(g(\theta_m)_{S_m^c}\big),$$

and trains on the enlarged support

$$A_m = S_m \cup G_m.$$

After the growth phase, the support is pruned by weight magnitude:

$$S_{m+1} = \operatorname{Top}_s(\theta_{m+1}).$$

Low-dimensional spectral phases. Let $\mathcal{M}_m$ denote the manifold of networks whose end-to-end map captures the first m singular modes,

$$W_2 W_1 \approx M_m := \sum_{i=1}^{m} s_i u_i v_i^\top.$$

In dense saddle-to-saddle dynamics, the trajectory moves through neighborhoods of $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_r$. Each $\mathcal{M}_m$ behaves as a saddle for the full problem: it is stable along already learned modes and unstable in directions that activate mode $m+1$.

Assumption C.9 (Sparse representation of the current saddle)**.** *For each m, there exists a support $S_m^\star$ of size at most s and a point $\theta_m^\star$ supported on $S_m^\star$ such that*

$$\|W_2^\star W_1^\star - M_m\|_F \le \delta_m.$$

Moreover, the support S_m produced by magnitude pruning satisfies

$$\|\theta_m - P_{S_m}\theta_m\|_2 \le \delta_m,$$

and S_m contains a support sufficient to represent M_m up to error δ_m.

Assumption C.10 (Residual dominance by the next singular mode)**.** *Near $\mathcal{M}_m$, the residual is dominated by the next singular component:*

$$E_m(\theta) := M - W_2 W_1 = s_{m+1} u_{m+1} v_{m+1}^\top + R_m(\theta),$$

with

$$\|R_m(\theta)\|_F \le \eta_m s_{m+1}, \qquad \eta_m < 1.$$

Assumption C.11 (Sparse escape direction)**.** *Let $e_{m+1}(\theta)$ denote the dense escape component of the gradient that increases the learned strength of singular mode $m+1$. The enlarged support*

$$A_m = S_m \cup G_m$$

captures this escape direction up to error ε_m:

$$\|(I - P_{A_m})e_{m+1}(\theta)\|_2 \le \varepsilon_m \|e_{m+1}(\theta)\|_2.$$

Assumption C.12 (No escape on the old support)**.** *The old support S_m has negligible projection onto the next escape direction:*

$$\|P_{S_m} e_{m+1}(\theta)\|_2 \le \alpha_m \|e_{m+1}(\theta)\|_2,$$

where $\alpha_m \ll 1$.

Assumption C.12 formalizes the need for rewiring: training only on S_m can consolidate the current mode, but it cannot efficiently move toward the next singular direction. Assumption C.11 states that the top-gradient growth set G_m supplies the missing coordinates needed to escape.

Theorem C.13 (Alternating sparse phases implement sparse saddle-to-saddle dynamics)**.** *Suppose Assumptions C.9–C.12 hold for phases $m = 0, \dots, r-1$. Assume also that $\mathcal{L}$ is L-smooth in a neighborhood of the relevant trajectories.*

Then each alternating cycle has the following behavior.

1. ***Consolidation.*** *Training on S_m decreases the loss primarily by optimizing the sparse representation of the already active modes $1, \dots, m$, and approaches a neighborhood of the sparse rank-m saddle $\mathcal{M}_m$.*
2. ***Stalling on the old support.*** *Near $\mathcal{M}_m$, the restricted gradient on S_m has only an α_m-fraction of the escape direction toward mode $m+1$. Hence training only on S_m leaves the trajectory near the rank-m sparse saddle.*
3. ***Sparse escape by rewiring.*** *Training on*

   $$A_m = S_m \cup G_m$$

   recovers a $1 - \varepsilon_m^2$ fraction of the dense escape-direction energy:

   $$\|P_{A_m} e_{m+1}(\theta)\|_2^2 \ge (1 - \varepsilon_m^2)\|e_{m+1}(\theta)\|_2^2.$$

   Therefore the enlarged sparse phase has a nontrivial component in the unstable direction that increases the learned strength of singular mode $m+1$.
4. ***Support transition.*** *After the growth phase, magnitude pruning preserves the large coordinates created by learning mode $m+1$. Hence the next support S_{m+1} represents M_{m+1} up to error δ_{m+1}.*

Consequently, the alternating sparse trajectory shadows the dense saddle-to-saddle sequence

$$\mathcal{M}_0 \to \mathcal{M}_1 \to \cdots \to \mathcal{M}_r,$$

up to perturbation errors controlled by δ_m, η_m, α_m, ε_m.

Proof. We prove the claim phase by phase.

Step 1: Consolidation on S_m. By Assumption C.9, the support S_m contains a sparse parameterization of the first m singular modes. Therefore the restricted flow

$$\dot{\theta} = -P_{S_m} g(\theta)$$

can reduce the component of the loss associated with the currently active modes. Near M_m, the residual is

$$E_m(\theta) = M - W_2 W_1 \approx \sum_{i>m} s_i u_i v_i^\top.$$

By Assumption C.10, the dominant remaining residual is the $(m+1)$-st singular component.

Step 2: The old support lacks the next escape direction. Let $e_{m+1}(\theta)$ be the dense gradient component that increases the learned strength of mode $m+1$. Assumption C.12 gives

$$\|P_{S_m} e_{m+1}(\theta)\|_2 \le \alpha_m \|e_{m+1}(\theta)\|_2.$$

Thus the old support contains little of the unstable direction needed to leave the rank-m saddle. Training on S_m can refine the sparse representation of the current modes, but it cannot efficiently activate mode $m+1$. This establishes the stalling behavior.

Step 3: The growth support recovers the escape direction. The growth set is chosen as

$$G_m = \mathrm{Top}_q(g(\theta_m)_{S_m^c}).$$

Since G_m contains the largest gradient coordinates outside the old support, and since the next-mode escape component dominates the residual gradient by Assumption C.10, Assumption C.11 implies

$$\|(I - P_{A_m}) e_{m+1}(\theta)\|_2 \le \varepsilon_m \|e_{m+1}(\theta)\|_2.$$

Therefore

$$\|P_{A_m} e_{m+1}(\theta)\|_2^2 = \|e_{m+1}(\theta)\|_2^2 - \|(I - P_{A_m}) e_{m+1}(\theta)\|_2^2 \ge (1 - \varepsilon_m^2)\|e_{m+1}(\theta)\|_2^2.$$

Thus the enlarged support A_m contains a substantial component of the unstable direction leading from $\mathcal{M}_m$ toward $\mathcal{M}_{m+1}$.

Step 4: Sparse escape increases the next mode. Let $q_{m+1}(\theta)$ denote the learned strength of mode $m+1$, for example

$$q_{m+1}(\theta) = u_{m+1}^\top W_2 W_1 v_{m+1}.$$

Under dense gradient flow, the derivative of q_{m+1} is positive near $\mathcal{M}_m$ because the residual is dominated by $s_{m+1} u_{m+1} v_{m+1}^\top$. Under sparse-growth flow,

$$\dot{\theta} = -P_{A_m} g(\theta).$$

The change in the mode strength is

$$\frac{d}{dt} q_{m+1}(\theta) = -\langle \nabla q_{m+1}(\theta), P_{A_m} g(\theta) \rangle.$$

Since $P_{A_m} g(\theta)$ captures a $1 - \varepsilon_m^2$ fraction of the escape direction, the sparse-growth phase preserves the positive mode-growth velocity up to an $O(\varepsilon_m)$ perturbation. Hence the trajectory moves away from $\mathcal{M}_m$ in the direction of $\mathcal{M}_{m+1}$.

Step 5: Magnitude pruning consolidates the new mode. During the sparse-growth phase, coordinates that contribute to mode $m+1$ receive coherent updates and grow in magnitude. Coordinates not aligned with the first $m+1$ modes either receive smaller updates or are not repeatedly reinforced. Hence magnitude pruning after the growth phase preserves a support S_{m+1} that represents

$$M_{m+1} = \sum_{i=1}^{m+1} s_i u_i v_i^\top$$

up to error δ_{m+1}, by Assumption C.9 applied at the next phase.

Repeating this argument for $m = 0, \dots, r-1$ gives the claimed sparse saddle-to-saddle sequence. □

D Training details

Tables 5 and 6 provide the default hyperparameters for ResNet18/CIFAR100 and DeiT-Small/ImageNet. The CIFAR100 experiments use cross-entropy loss. The DeiT-Small experiments use label smoothing with confidence 0.9 and CutMix augmentation [75]. These experiments use three seeds (1, 2, and 3). For all vision experiments conducted by us, we report means and 95% normal-approximation confidence intervals over the three seeds. For all ARC experiments, masked parameters are restored to their values saved at the end of the preceding dense or rewiring phase (Stale init), or reset to zero (Zero init), before computing the gradients used to select G_t. The implementation is based on TurboPrune [50], and each run uses four NVIDIA A100 GPUs with 40GB memory. The additional ResNet50 and language-model settings are described below.

Table 5: Default training hyperparameters for ResNet18/CIFAR100 and DeiT-Small/ImageNet.

Configuration	**CIFAR100 Experiment**	**ImageNet Experiment**
Dataset	CIFAR100	ImageNet
Model	ResNet18	DeiT-small
LR	0.1	0.05
WD	$1e-4$	$1e-1$
Epochs	150	300
Batch Size	128	256
Optimizer	SGD	AdamW
β_1	–	0.9
β_2	–	0.999
Schedule	Triangular	Triangular

Table 6: Default AC/DC, ARC, and RigL settings for ResNet18/CIFAR100 and DeiT-Small/ImageNet. Shortened-phase configurations are specified separately in the corresponding experiments.

Configuration	**CIFAR100 Experiment**	**ImageNet Experiment**
ACDC/ARC Dense warmup epochs	15	15
ACDC/ARC Dense epochs	10	15
ACDC/ARC Sparse epochs	10	15
ACDC/ARC Sparse cooldown epochs	15	15
ARC - Batches to compute G_t	5	5
RigL - Dense gradient accumulation steps	100	10
RigL - Rewiring steps	100	1000

D.1 FLOP accounting

We count one multiply–accumulate as two FLOPs. Vision training costs are calculated layer by layer from the active support in each phase and accumulated over the training schedule, including the gradient probes used for support selection. For CIFAR100 and DeiT-Small, we report epoch-averaged training FLOPs per image:

$$F_{\text{avg}} = \frac{F_{\text{total}}}{EN_{\text{train}}},$$

where E is the number of training epochs and N_{train} is the number of training images. For ResNet50, Table 3 reports total training FLOPs relative to the dense-training baseline. Inference FLOPs are calculated separately from the final support for one input image. Global sparsity can produce different layerwise allocations, so equal parameter sparsity need not imply equal inference FLOPs.

Support selection requires gradient probes over the eligible parameters, but is performed only at the beginning of each rewiring phase. The selected support remains fixed throughout that phase, so the selection cost is amortized over the subsequent training updates rather than incurred at every optimization step.

D.2 ResNet50 on ImageNet

We follow the ResNet50/ImageNet training recipe from the AC/DC paper [60]. The eligible weights are those in convolutional and fully connected layers, excluding the first convolution and the final classification head. For the unstructured experiments, sparse phases use 95% target sparsity. The 100-epoch ARC schedule consists of five dense warmup epochs, eight cycles of five sparse epochs followed by five rewiring epochs, and fifteen sparse cooldown epochs.

Each rewiring phase has 30.65% density across eligible weights: the inherited 5% support plus an additional 25.65 percentage points selected using off-support gradient magnitudes. Support selection uses five probe batches at each of the eight rewiring boundaries. The cost of all forty probe batches is included in the reported training FLOPs. ARC–Stale and ARC–Zero use 0.7289 and 0.7634 EFLOPs,

respectively, over the complete training run. Their final inference costs are 0.75437 and 0.8849 GFLOPs per image, respectively. Both ARC variants are evaluated over three seeds.

D.3 Language-model training and evaluation

Shared training recipe. We train GPT-2 124M [62] from scratch using nanoGPT [33]. The model has twelve Transformer blocks, twelve attention heads, hidden dimension 768, tied input and output embeddings, and no biases or dropout. All methods use seed 1337 and the same base training recipe. The context length is 1,024 tokens and the effective batch size is 480 sequences, or 491,520 tokens per optimizer update. Training follows a 600,000-update schedule using AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay 0.1 on matrix parameters, gradient clipping at 1.0, and BF16 mixed precision. The learning rate increases linearly to 6×10^{-4} over 2,000 updates, then follows cosine decay to 6×10^{-5}. It is not restarted at phase boundaries.

Data. We use OpenWebText [22] with the nanoGPT preprocessing pipeline. A document-level split holds out 0.05% of the corpus for validation using split seed 2357. Documents are tokenized using the GPT-2 byte-pair encoding tokenizer, separated by end-of-text tokens, and concatenated. Training samples random 1,024-token windows with replacement.

Sparsity and phase schedules. Pruning applies to the 48 attention and MLP weight matrices, containing 84,934,656 parameters. Embeddings, the tied output head, and normalization parameters remain dense. The reported 80% sparsity refers to these eligible matrices. AC/DC and ARC use 120,000 dense warmup updates, three cycles of 60,000 sparse and 60,000 dense or rewiring updates, and 120,000 sparse cooldown updates. At each sparse-phase boundary, magnitude pruning is applied globally across eligible weights.

For ARC, each rewiring phase reactivates the highest-gradient 25% of masked weights within each layer, giving approximately 40% overall density across eligible matrices. Before computing the selection gradients, masked weights are restored to their stale values or reset to zero according to the variant. Both ARC variants average gradients over 100 probe mini-batches. The inherited and selected supports remain fixed throughout the rewiring phase, and weights outside this support are kept at zero.

RigL starts from an 80%-sparse random mask with ERK layerwise allocation. It rewires every 100 updates until update 450,000, using a cosine-decayed rewiring fraction starting at 0.3. It removes the smallest-magnitude active weights and activates inactive weights with the largest gradient magnitudes. Newly activated weights and their Adam moments are initialized to zero. The mask remains fixed for the final 150,000 updates.

Evaluation. All reported metrics use the end-of-training checkpoint, without checkpoint selection by validation performance. OpenWebText validation loss and perplexity are evaluated on 4,330 non-overlapping windows of 1,024 tokens. Zero-shot evaluation uses lm-evaluation-harness version 0.4.3 [21] through a nanoGPT wrapper. The wrapper excludes padded vocabulary logits before scoring. We evaluate the full validation sets for HellaSwag, PIQA, and WinoGrande, and the full test sets for OpenBookQA, ARC-Easy, ARC-Challenge, and LAMBADA. LAMBADA uses the `lambada_openai` task. The reported average is the arithmetic mean of the seven task scores listed in Table 7.

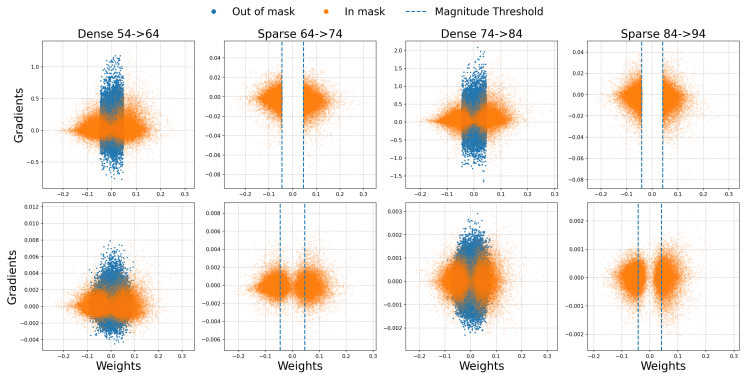
E Additional experiments

E.1 Small weights receive large gradients

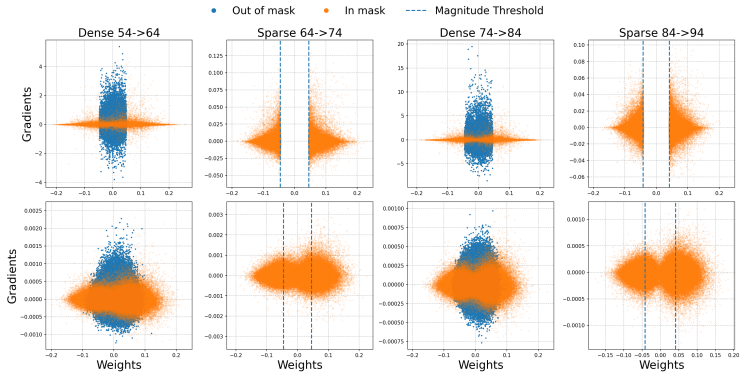
The observation that while sparse training causes parameter collapse, dense phase performs rewiring to recover relevant parameters can be observed across the layers of models (Figure 7) and even in AC/DC configurations with smaller dense phases (Figure 8).

E.2 Replacing the dense phase: gradient-selected versus random support

Figure 9 evaluates replacing AC/DC's dense phases with training restricted to a fixed random subset of parameters on ResNet18/CIFAR100. Random selection avoids the dense-gradient computation



(a) *layer2.0.conv2* of ResNet18



(b) *layer4.0.conv2* of ResNet18

Figure 7: Gradient vs parameter scatter plots of parameters of layers of a 90% sparse ResNet18 model on CIFAR100. The plots are shown for the 3rd and 4th dense and sparse phases of the 10-Dense, 10-Sparse schedule of ACDC with *Stale Init*. (top) plots are at the beginning of a phase and (bottom) plots are at the end of a phase.

required to identify the rewiring support, but yields lower accuracy than selecting parameters with the largest gradient magnitudes. This comparison supports using gradient information to guide support selection during rewiring.

E.3 Language-model evaluation results

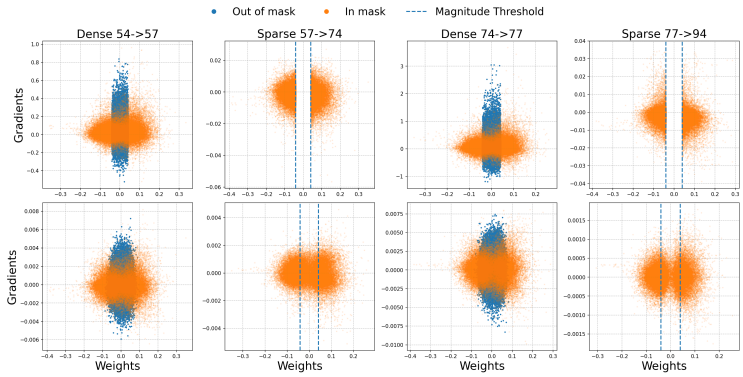
Table 7 reports the complete zero-shot evaluation. We use normalized accuracy (`acc_norm`) for OpenBookQA, ARC-Challenge, ARC-Easy, PIQA, and HellaSwag, and accuracy (`acc`) for WinoGrande and LAMBADA. ARC–Zero achieves the highest average among the sparse methods and improves over AC/DC on six of the seven tasks.

For context, the dense nanoGPT reproduction [33] reports an OpenWebText validation loss of approximately 2.85, corresponding to perplexity 17.3. Our evaluation of the released OpenAI GPT-2 checkpoint [62] gives validation loss 3.11387 and perplexity 22.5081. These are external dense reference points rather than matched training baselines; the OpenAI checkpoint was trained on WebText, not OpenWebText.

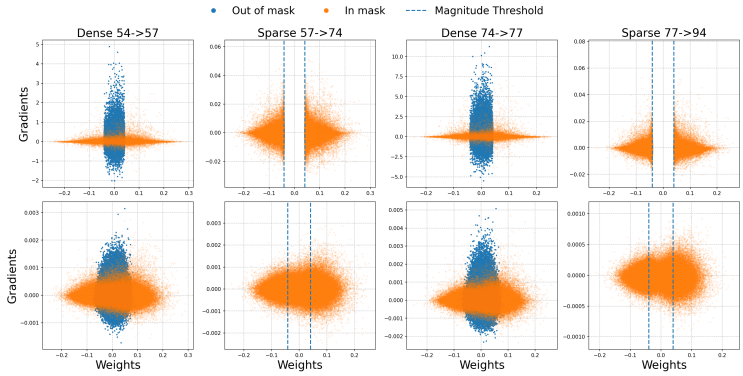
E.4 2:4 semi-structured sparsity

We evaluate a 2:4 semi-structured extension on ResNet50/ImageNet using the same base training recipe, phase schedule, and eligible layers as the unstructured ResNet50 experiments in Appendix D.2. The first convolution and final classification head remain dense.

During sparse phases, eligible convolutional weights follow a 2:4 pattern. For each output channel and kernel position (r, s), groups consist of four consecutive input channels. Magnitude pruning



(a) *layer2.0.conv2* of ResNet18



(b) *layer4.0.conv2* of ResNet18

Figure 8: Gradient vs parameter scatter plots of parameters of layers of a 90% sparse ResNet18 model on CIFAR100. The plots are shown for the 3rd and 4th dense and sparse phases of the 3-Dense, 17-Sparse schedule of ACDC with *Stale Init*. (top) plots are at the beginning of a phase and (bottom) plots are at the end of a phase.

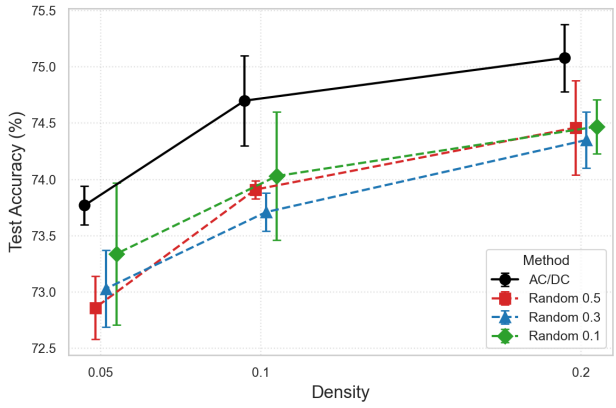


Figure 9: Replacing dense phases in AC/DC with updating only a random subset of parameters for CIFAR100 trained on ResNet18.

retains two weights in each group. This grouping follows NVIDIA's documented convolutional structured-sparsity convention [52].

At the beginning of each ARC rewiring phase, we use five probe mini-batches to score output channels by their off-mask gradient energy, separately within each layer. The highest-scoring half of the output channels are temporarily made dense, while the remaining channels retain their existing

Table 7: **Zero-shot evaluation of GPT-2 124M.** Scores are percentages from one training seed (1337). Bold indicates the best score among the four sparse methods. The final column evaluates the released dense OpenAI GPT-2 checkpoint as an external reference.

Task	AC/DC	ARC–Stale	ARC–Zero	RigL	Dense GPT-2 reference
OpenBookQA [48]	26.80	**27.00**	26.20	25.40	26.40
ARC-Challenge [9]	23.21	23.38	**23.55**	22.10	22.10
ARC-Easy [9]	38.05	37.63	**39.23**	37.24	40.24
WinoGrande [64]	50.28	48.86	**52.64**	50.05	51.85
PIQA [4]	59.09	59.90	**61.59**	60.34	62.40
HellaSwag [76]	29.24	29.67	**29.76**	28.75	31.50
LAMBADA [54]	27.23	28.00	**28.06**	25.83	30.74
Average	36.27	36.35	**37.29**	35.67	37.89

Table 8: **2:4 semi-structured ResNet50 on ImageNet.** Top-1 accuracy (%) is reported as the mean and 95% normal-approximation confidence interval over three seeds. ARC uses 75% density in eligible layers during rewiring; AC/DC uses fully dense decompression.

Method	Top-1 accuracy (%)
AC/DC–Stale	76.29 ± 0.06
ARC–Stale	75.96 ± 0.14
AC/DC–Zero	76.41 ± 0.07
ARC–Zero	76.06 ± 0.07

2:4 support. This gives 75% parameter density in eligible layers during rewiring, rather than fully dense decompression. The selected channel partition remains fixed throughout the phase. Before the following sparse phase, the temporarily dense channels are pruned back to 2:4 using weight magnitude.

Table 8 reports results over three seeds with 95% normal-approximation confidence intervals. ARC–Stale and ARC–Zero are within 0.33 and 0.35 percentage points of their corresponding AC/DC variants. Thus, targeted support expansion also retains accuracy close to AC/DC under a semi-structured sparsity constraint. These experiments evaluate accuracy; they do not establish measured wall-clock acceleration.

F Additional Activation Eigenspace Diagnostics

This appendix provides the construction and additional results for the activation-subspace diagnostic in Section 3.3.

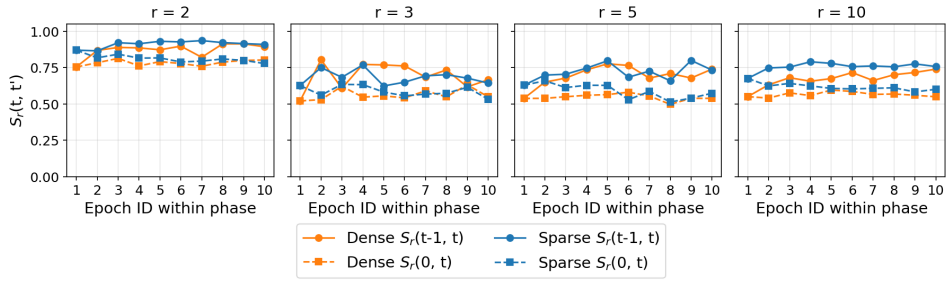
Activation Gram matrix. For a fixed convolutional layer, let $Z_t(x) \in \mathbb{R}^{C\times H\times W}$ denote its activations for image x at checkpoint t, where C is the number of output channels and H, W are the spatial feature-map dimensions. Flattening the spatial dimensions gives $F_t(x) \in \mathbb{R}^{C\times HW}$. We aggregate the per-image Gram matrices over the CIFAR100 training set $\mathcal{D}$ before computing the eigenvectors:

$$K_t = \frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} F_t(x)F_t(x)^\top \in \mathbb{R}^{C\times C}. \tag{2}$$

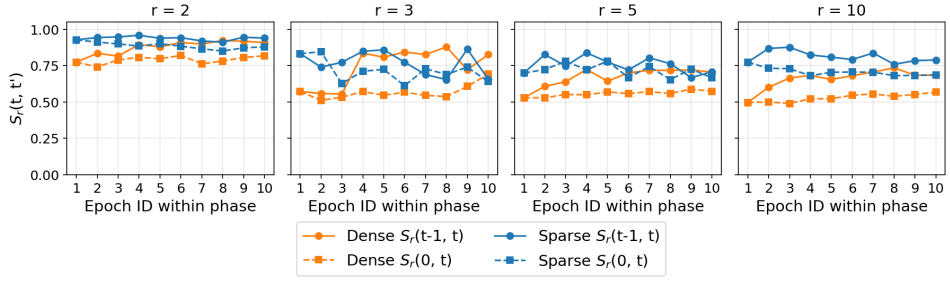
Summing instead of averaging multiplies K_t by a positive scalar and therefore leaves its eigenspaces unchanged. Each entry measures the average inner product between two channels' spatial activation maps. The leading eigenvectors identify channel combinations with the greatest activation energy.

Subspace comparison. Let $U_t^{(r)} \in \mathbb{R}^{C\times r}$ contain orthonormal eigenvectors corresponding to the r largest eigenvalues of K_t. We define

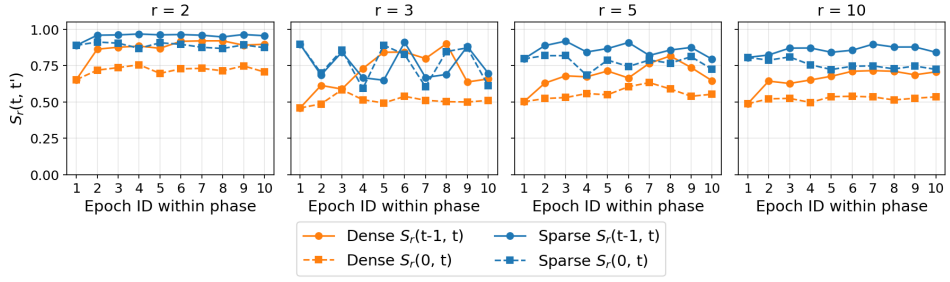
$$S_r(t_1, t_2) = \frac{1}{r}\left\| {U_{t_1}^{(r)}}^\top U_{t_2}^{(r)} \right\|_F^2.$$



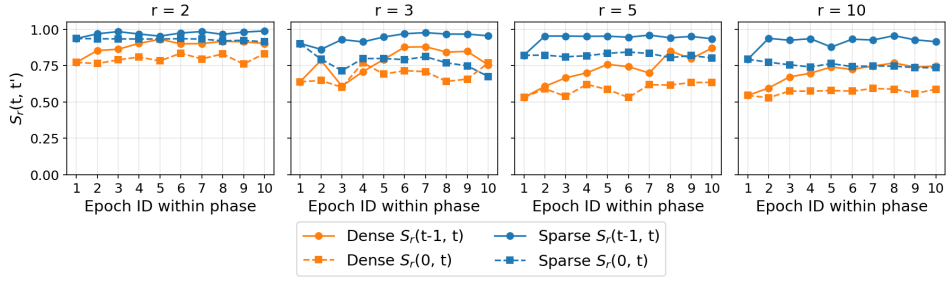
(a) First cycle (Sparse:24-34, Dense:34-44)



(b) Second cycle ((Sparse:44-54, Dense:54-64))



(c) Third cycle ((Sparse:64-74, Dense:74-84))



(d) Fourth cycle ((Sparse:84-94, Dense:94-104))

Figure 10: **Activation subspace similarity across the first four AC/DC cycles.** For *layer3.1.conv2* of a 90% sparse ResNet18 trained on CIFAR100, we compare the subspaces spanned by the top $r \in \{2, 3, 5, 10\}$ activation Gram eigenvectors. We report consecutive-checkpoint similarity $S_r(t-1, t)$ and similarity to the first checkpoint of each phase $S_r(0, t)$. Sparse phases consistently preserve leading activation subspaces more strongly than dense phases, especially for $r = 5$ and $r = 10$. Dense phases induce larger but smooth drift, consistent with their role as rewiring windows.

This is the average squared cosine of the principal angles between the two subspaces. It lies in $[0, 1]$, equals one for coincident subspaces, and equals zero for orthogonal subspaces. Comparing subspaces avoids dependence on individual eigenvector signs or the choice of orthonormal basis within a retained subspace. We report consecutive-checkpoint similarity $S_r(t-1, t)$ and similarity to the first checkpoint of each phase, $S_r(0, t)$. The time index is reset at the beginning of each phase.

We additionally show individual-eigendirection similarity heatmaps to examine changes within the leading spectrum.

Figure 10 reports this metric for the first four AC/DC cycles of *layer3.1.conv2*. Across cycles, sparse phases preserve leading activation subspaces more strongly than dense phases. This difference is most visible for $r = 5$ and $r = 10$. Later cycles show especially high sparse-phase similarity, consistent with increasing representation stability over training. Dense phases show larger drift from the phase start, but consecutive-epoch similarities remain substantial, indicating gradual representation movement rather than abrupt resets.

Figures 11, 12 provide the eigendirection cosine-similarity heatmaps for dense and sparse phases of the first and fourth cycles. These heatmaps show structured alignment among leading eigendirections, with greater mixing among lower-ranked directions. They provide a more detailed view of the subspace trends summarized in the main text.

G Final-Mask Sign Flips Occur During Rewiring Windows

Prior work on lottery-ticket-style sparse training shows that surviving weights are characterized not only by their mask but also by their signs [16, 18]. Sign-In to the Lottery further studies sign correction as a mechanism for improving sparse training [19]. We use the timing of the last sign flip to identify when parameters undergo persistent directional corrections.

Counting protocol. A sign flip is a reversal between two nonzero parameter values. During dense phases, each parameter is compared against its last active nonzero value, carrying this reference across masked or zero-valued intervals. During sparse phases, active parameters are compared between consecutive checkpoints; pruned parameters cannot flip. Pruning and zero initialization are not themselves counted as sign reversals.

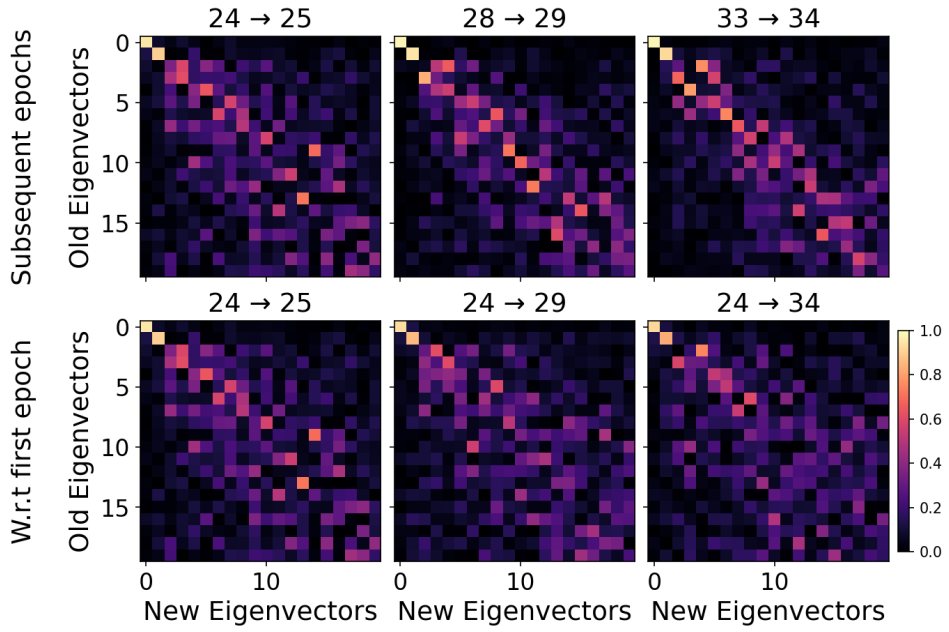
Each parameter is assigned to the phase containing its last sign flip. Parameters that never flip are assigned to the warmup bin, which therefore represents parameters whose signs have settled by the end of warmup. We apply this analysis separately to parameters retained in the final sparse mask and to parameters active at every intermediate checkpoint, for both Zero init and Stale init. For the final-mask plots, phase counts are normalized by the final mask size.

Additional AC/DC diagnostics. The main aggregate comparison is shown in Figure 2a. Most final-mask parameters settle their signs by the end of warmup, while later sign corrections occur mainly during dense phases. Stale initialization concentrates these corrections in early dense phases; zero initialization continues to produce corrections later in training.

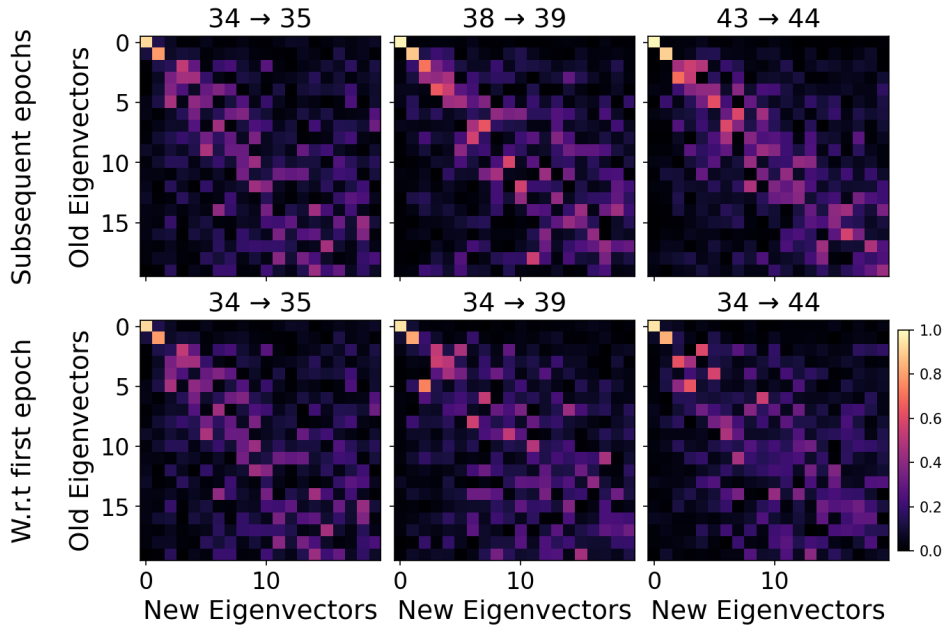
Figure 13 compares the final-mask and always-active parameter sets under shortened dense phases. Figure 14 shows the corresponding layerwise analysis. The ability of Zero init to produce meaningful sign flips later in training persists across layers and shortened-phase configurations. Together with its stronger performance in Figure 4, these findings indicate that Zero init provides better directional correction.

Sign correction under ARC. Figure 2b applies the same final-mask sign-flip diagnostic to ARC. Late sign corrections occur primarily during ARC rewiring phases rather than sparse phases. Thus, replacing dense phases with sparse updates on $A_t = S_t \cup G_t$ preserves the phase-wise correction pattern observed in AC/DC, but compresses it onto a smaller active support.

The comparison between Stale init and Zero init also changes under ARC. In AC/DC, Stale init concentrates most late sign corrections in early dense phases. Under ARC, Stale init can maintain nontrivial sign correction across later rewiring phases. This suggests that restricting updates to high-gradient reintroduced coordinates changes the reactivation dynamics while preserving the correction role of expanded phases.

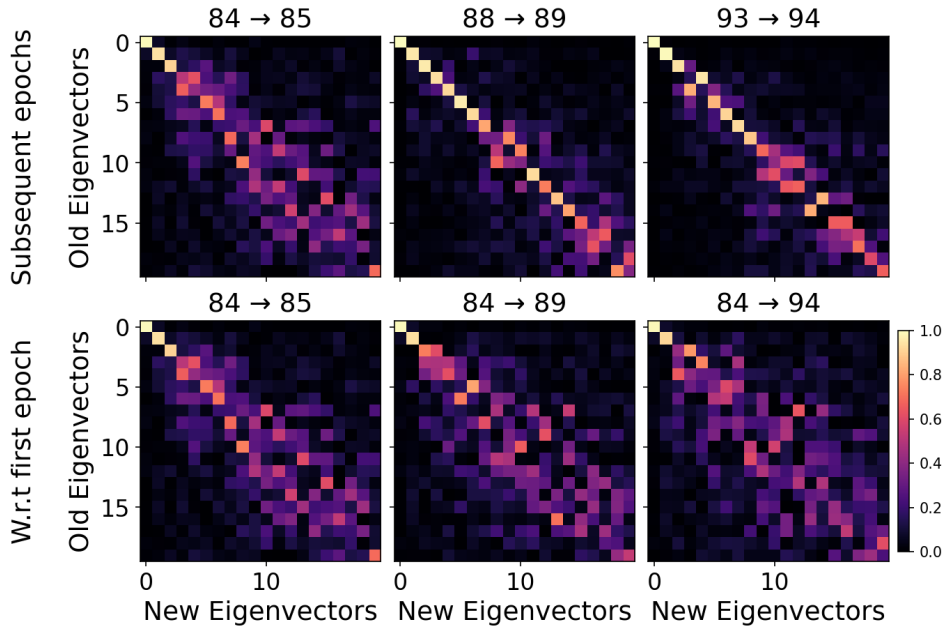


(a) Sparse:24-34

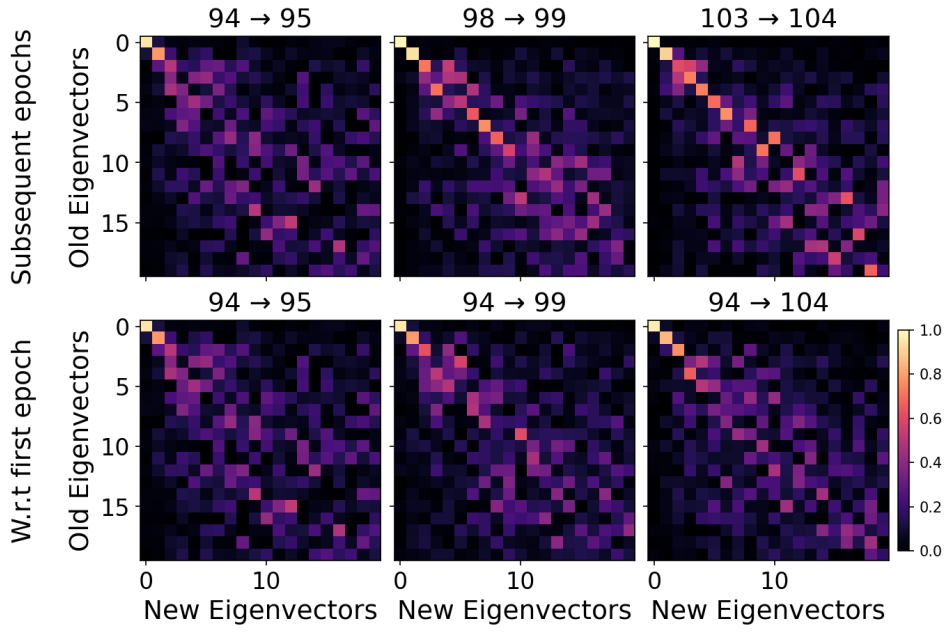


(b) Dense:34-44

Figure 11: **Activation eigendirection similarities in the first AC/DC cycle.** We compare activation-Gram eigenvectors for *layer3.1.conv2* of a 90% sparse ResNet18 on CIFAR100 during the sparse phase (epochs 24–34) and dense phase (epochs 34–44). Within each panel, the top row compares consecutive checkpoints and the bottom row compares checkpoints with the phase start. Rows and columns index eigenvectors at the earlier and later checkpoints, respectively.

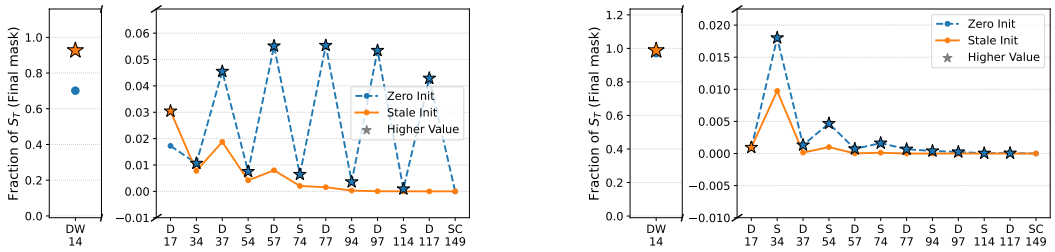


(a) Sparse:84-94



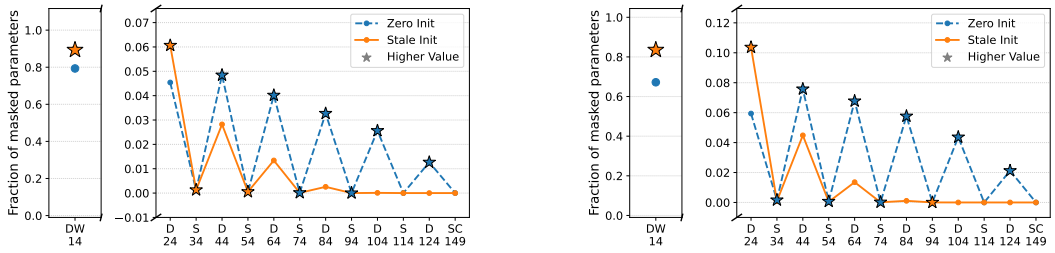
(b) Dense:94-104

Figure 12: **Activation eigendirection similarities in the fourth AC/DC cycle.** We compare activation-Gram eigenvectors for *layer3.1.conv2* of a 90% sparse ResNet18 on CIFAR100 during the sparse phase (epochs 84–94) and dense phase (epochs 94–104). Within each panel, the top row compares consecutive checkpoints and the bottom row compares checkpoints with the phase start. Rows and columns index eigenvectors at the earlier and later checkpoints, respectively.



(a) (b)

Figure 13: The plot shows the fraction of weights of ResNet18 that flip their sign for the last time during a specific phase of AC/DC (3-dense, 17-sparse). Panels compare sign convergence for parameters in (a) the final converged mask and (b) the stable core weights found in the intersection of all masks.



(a) *layer2.0.conv2* of ResNet18 (b) *layer4.0.conv2* of ResNet18

Figure 14: The plot shows the fraction of weights of specific layers of ResNet18 that flip their sign for the last time during a specific phase of AC/DC (10-dense, 10-sparse). Panels compare sign convergence for parameters of layers *layer2.0.conv2*(a) and *layer4.0.conv2* in the final converged mask.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer:[Yes]

   Justification: We set out to improve the performance-to-FLOP frontier with 3 innovations which we present in the main part of the manuscript.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are explained in the sections "End-to-End Performance and FLOP Tradeoffs" and Conclusion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Detailed proofs and full assumptions for each statement in the main text are provided in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The algorithmic innovations are described in detail in the main text. The setup details and underlying hyperparameter choices are given in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code used to conduct the experiments is attached as a ZIP file.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: The details needed for reproducing the experiments are provided in the appendix such, as base optimizer, lr and batchsize choices.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: No human subjects where used in the work of this manuscript.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer:[Yes]

   Justification: We read and adhere to the code of ethics and have made sure to preserve anonymity.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our work furthers the general understanding of sparse training which puts the manuscript in the foundational category. Therefore there are no direct negative social impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our algorithms are tested only general standard public benchmarks. We dont release any of the aforementioned items.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We credit the datasets and previously proposed relevant algorithms in the main text.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The code used for the development of the general applicable sparse training method is attached to the manuscript. Moreover, the details of the algorithmic inovations is described in detail in the main text.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No human subjects where used in the work of this manuscript.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human subjects where used in the work of this manuscript.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: An LLM has not been used for any non-standard component of this manuscript.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.