How Do Weight and Activation Sparsity Interact?

Anonymous authors
Paper under double-blind review

Abstract

A multiply-accumulate (MAC) contributes to inference cost only if *both* its weight and activation operand are nonzero, yet the two are almost always regularized independently. We test whether jointly regularizing both, with a shared per-element-normalized squared-Hoyer measure, causes them to reinforce or compete, and find the answer is conditional rather than universal. On ResNet-18/CIFAR-10, joint (additive) training pushes both sparsities above what either solo penalty reaches alone (weight sparsity 86.75% vs. 86.08%; activation sparsity 67.72% vs. 63.08%; $n{=}5$ seeds), and the resulting *effective-MAC fraction* beats a naive independence estimate at a $10.5\times$ reduction and 97.0% of baseline accuracy – though at matched compute a solo penalty alone reaches statistically indistinguishable accuracy, so the gain is a different sparsity split, not a proven accuracy advantage. A theoretically appealing *coupled* (product) objective instead underperforms the additive sum by 14.1 accuracy points at matched compute, substantially a convergence-speed artifact: the gap shrinks by more than half after doubling the training budget and closes to 2.2 points by 50 epochs. Reinforcement replicates in most but not all of six further settings (MNIST, CIFAR-100, SVHN, MobileNetV2, and structured sparsity) – reversing on MNIST, an order of magnitude stronger under structured sparsity, and turning outright bimodal on SVHN – and the same conditional pattern holds under an entirely different, discrete weight-sparsity mechanism on four independent production edge-AI models, where the interaction is complementary, antagonistic, sign-flipping, or accuracy-improving depending on the model. Critically, the coupled objective's failure is a fact about Hoyer specifically, not product-form coupling in general: a density-aligned surrogate closes a 14.2-point accuracy gap to 0.3 points, statistically indistinguishable from additive. We report this as evidence that weight and activation sparsity interact dynamically and conditionally rather than independently.

1 Introduction

Weight pruning and activation sparsity are usually studied as separate research programs, even though both target the same quantity: a multiply-accumulate $y \mathrel{+}= w_i \cdot a_i$ contributes nothing, and can in principle be skipped, only if *either* $w_i = 0$ *or* $a_i = 0$. Prior work has jointly *trained* for both (Section 2) as an engineering technique, not as a test of whether the two pressures reinforce or compete, or whether a product-form objective surrogating this AND-condition helps or hurts. An additive regularizer, $\lambda_w H(W) + \lambda_a H(A)$, rewards each margin separately; it does not reward eliminating a MAC through *either* route the way the OR-condition allows. A product-form regularizer, $H(W) \cdot H(A)$, is a closer tensor-level analogue: since $H \in [0, 1]$, driving the product to zero requires only one factor to shrink – though it remains a per-layer, per-tensor surrogate for a condition that is actually defined per weight-activation pair (Proposition 4 makes this gap between marginal and pairwise structure explicit). This framing raises two questions: (1) does jointly regularizing weight and activation sparsity reinforce or compete, relative to either alone? (2) does the coupled (product) objective outperform the additive one in practice?

There is a plausible mechanistic reason to expect reinforcement, which we state as two exact propositions identifying mechanisms consistent with it – not a predictive theory of when it occurs, nor a proof that it must. Let $z = \sum_i w_i a_i$ be a unit's pre-activation feeding $y = \mathrm{ReLU}(z)$ and a downstream loss L.

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

Proposition 1 (gradient shielding). *For any example on which $a_i = 0$, $\partial L/\partial w_i = 0$ exactly (proof in Appendix A).* Whenever activation sparsity zeroes a_i, only the weight regularizer's own gradient acts on w_i for that example – a necessary condition for reinforcement, not a sufficient one: it drives sparsity further when w_i's task-optimal value is itself zero, but suppresses it when the task genuinely needs $w_i \neq 0$ (Appendix B makes this double edge concrete).

Proposition 2 (variance shrinkage). *For independent a_i with $\mathrm{Var}(a_i) = \sigma_i^2$, $\mathrm{Var}(z) = \sum_i w_i^2 \sigma_i^2$, which weakly decreases as any $|w_i| \to 0$ (proof in Appendix A).* Weight sparsity therefore shrinks the pre-activation's variance, making it easier for ReLU to zero (direct causal test: Appendix Q).

Together these describe two directional halves of a plausible feedback loop, not sufficient conditions for it: we do not attempt a fixed-point analysis of the full joint dynamic, and predicting the sign of the weight-activation interaction in a real trained network from first principles alone remains open. A minimal two-parameter toy model shows the mechanism is double-edged: joint regularization provably reaches exact joint sparsity a solo penalty structurally cannot on a redundant unit, but can just as easily suppress sparsity on a unit the task needs (Appendix B). We test the loss-level question with five regimes on ResNet-18/CIFAR-10 – baseline, weight-only, activation-only, joint additive, and joint coupled – via an explicit **effective-MAC fraction** validated against exact instrumentation, then test generality across architecture, task difficulty, domain, and sparsity granularity, and diagnose the coupled objective's underperformance. Our central findings: reinforcement holds in most settings but is conditional – weaker on some marginal sparsities than others, weaker under milder calibration, and reversed in aggregate on one architecture – and the coupled objective's disadvantage is substantially a convergence-speed artifact rather than a fixed calibration or optimum problem, though it also interacts destructively with task difficulty and, on one dataset, degenerates into outright bimodal instability.

2 Related Work

Hardware accelerators that skip multiply-accumulates with a zero operand, such as EIE (Han et al., 2016), are the practical motivation for treating weight and activation sparsity as jointly compute-relevant rather than independently interesting quantities – exactly the effective-MAC accounting this paper formalizes (Section 3.2).

The Hoyer measure, the ratio of L_1 to L_2 norms, was introduced for sparse non-negative matrix factorization (Hoyer, 2004). DeepHoyer (Yang et al., 2020) popularizes a differentiable squared form and applies it to weight and activation tensors separately, including a structured "Group-HS" variant, but never combines the two into a joint objective, and uses the raw, unnormalized ratio, without the per-element normalization we argue (Section 3.1, Proposition 3) is necessary for one coefficient to exert comparable pressure across differently-sized layers. Classical weight pruning (Han et al., 2015) and activation-sparsity-for-inference work (Kurtz et al., 2020) each optimize one operand in isolation.

Joint weight-and-activation regularization as an engineering technique does exist: JPnet (Yang et al., 2019) trains with an *additive* loss, $\alpha \sum_l \|W_l\|_1 + \beta \sum_l \|A_l\|_0$ (the same additive form as our joint-additive regime, applied to different penalty functions), reporting large real MAC reductions across MLP, CNN, and ResNet architectures; SWAT (Raihan & Aamodt, 2020) sparsifies both weights and activations during training itself, primarily to accelerate training rather than to study the resulting interaction; and ARTS (Zhu et al., 2022) adaptively schedules an activation-sparsity coefficient over training. None of these ask whether joint training pushes either sparsity beyond what the corresponding solo penalty reaches alone, measure any statistical association between which weights and which activations end up zero, or test a coupled (product) formulation and diagnose why it might fail – the three central questions this paper investigates. JPnet's own results are, if anything, a data point for interaction in the *opposite* direction from our reinforcement finding: it reports that weight pruning alone *increases* the non-zero activation percentage relative to a dense baseline (competition), the kind of marginal-sparsity comparison this paper makes systematically, with matched solo baselines, significance testing, and both additive and coupled objectives.

Structured pruning – group-Lasso-regularized structured sparsity (Wen et al., 2016), filter pruning (Li et al., 2017), network slimming (Liu et al., 2017), and channel pruning (He et al., 2017) – removes whole output channels, mechanically zeroing both that channel's weights and every downstream activation depending on it, an implicit coupling as a structural byproduct; none report or analyze whether the resulting sparsities reinforce or compete relative to what either would reach alone, the explicit, unstructured, directly-measured comparison this paper makes (and also tests under structured

granularity directly, Section D.6). Hardware-aware N:M fine-grained structured sparsity (Zhou et al., 2021) sits at a granularity between our unstructured and channel-level cases, an open direction for the reinforcement question this paper asks.

A separate line of work varies weight sparsity *during* training rather than fixing a differentiable penalty throughout: the lottery ticket hypothesis (Frankle & Carbin, 2019) finds that a dense network contains sparse subnetworks trainable from their original initialization, and RigL (Evci et al., 2020) maintains a fixed sparsity budget while dynamically reallocating which weights are pruned and regrown based on magnitude and gradient information. Neither varies activation sparsity jointly or asks whether the two interact; we instead use a fixed differentiable regularizer on both operands simultaneously throughout training, trading the ability to reach an exact target sparsity for a continuous signal that lets us measure and diagnose the interaction directly, including its failure mode under a coupled objective. That density surrogate (Section 4.2) is itself conceptually related to the hard-concrete relaxation Louizos et al. use for differentiable L_0 regularization (Louizos et al., 2018), and to variational/Bayesian sparsification more broadly (Molchanov et al., 2017) – all replace a non-differentiable zero-or-not count with a smooth, density-aligned or probabilistic proxy – though we use it to test a specific hypothesis about why product-form coupling fails under Hoyer, rather than as a standalone sparsity penalty.

3 METHOD

3.1 NORMALIZED HOYER MEASURES

For an activation tensor A_l we use the per-sample, per-element-normalized squared-Hoyer measure

$$H(A_l) = \mathbb{E}_b \left[\frac{\left(\sum_i |a_{b,i}|\right)^2}{\sum_i a_{b,i}^2 + \epsilon} \Big/ N_A \right], \tag{1}$$

and for a weight tensor W_l, $H(W_l) = \left(\sum_i |w_i|\right)^2 / (\sum_i w_i^2 + \epsilon)/N_W$. Both are bounded in $[0, 1]$ regardless of tensor size N ($H = 0$ exactly only for the all-zero tensor, since ϵ in the denominator keeps H well-defined there; for any nonzero tensor $H \in [1/N, 1]$, per Proposition 3 below). This normalization matters for using one shared coefficient across differently-sized layers: without it, a fixed coefficient exerts pressure on the raw ratio $\rho(x) = (\sum_i |x_i|)^2 / \sum_i x_i^2$ that varies with layer size.

Proposition 3 (range of the raw ratio). *For $x \in \mathbb{R}^N \setminus \{0\}$, $1 \le \rho(x) \le N$, with the lower bound tight iff at most one coordinate is nonzero and the upper bound tight iff all nonzero coordinates have equal magnitude (proof in Appendix A).* Dividing by N collapses this layer-size-dependent range to $[1/N, 1] \approx (0, 1]$.

3.2 FIVE REGIMES AND THE EFFECTIVE-MAC FRACTION

We compare baseline ($\mathcal{L}_{\text{CE}}$), weight-only ($+\lambda_w \sum_l H(W_l)$), activation-only ($+\lambda_a \sum_l H(A_l)$), joint additive ($+\lambda_w \sum_l H(W_l) + \lambda_a \sum_l H(A_l)$), and joint coupled ($+\lambda_c \sum_l H(W_l) \cdot H(A_l)$). Since a residual network has more convolutional weight tensors than hooked activation tensors, the coupled term and the effective-MAC accounting below pair each weight tensor with its nearest preceding activation by relative depth.

Per layer, $C_l = (1-\text{sparsity}(W_l))\cdot(1-\text{sparsity}(A_l))$, using measured (not differentiable) sparsity, under an independence assumption between zero-weight and zero-activation locations; $C = \text{mean}_l(C_l)$ is the headline effective-compute metric (lower is better). This independence assumption is one term of an exact identity we verify directly (Section 4.4, Appendix K). C measures the fraction of multiply-accumulates that remain under the observed joint sparsity pattern – a proxy for arithmetic opportunity, not a direct prediction of wall-clock latency or energy, which additionally depend on sparse-kernel efficiency, indexing overhead, and memory bandwidth that we do not measure here.

Proposition 4 (exact decomposition). *Let X, Y be $\{0, 1\}$-valued indicators for "weight/activation operand nonzero" over layer l's real (weight, activation) operand pairs. Then $C_l^{exact} = P(X{=}1)P(Y{=}1) + \text{Cov}(X, Y)$ (proof in Appendix A).* This separates two effects reinforcement claims otherwise conflate: joint training raising the marginals changes C_l through the independence

term alone; only a nonzero covariance reflects genuine within-layer statistical association, which we measure directly in Appendix K.

3.3 Architecture, Training, and Calibration

We use a CIFAR-style ResNet-18 (20 `Conv2d` weight tensors, 17 hooked activations), Adam at $\text{lr} = 10^{-3}$ with cosine annealing, batch size 128, standard augmentation, 10 epochs and three seeds unless noted. A two-stage coefficient sweep (screening at 5 epochs over a grid, then full-protocol validation of the Pareto-optimal candidates; Appendix C) selects $\lambda_w = 0.3$, $\lambda_a = 0.05$ for joint additive and $\lambda_c = 0.8$ for joint coupled (matched to additive's effective-MAC fraction at a cheap 5-epoch proxy) as the primary configuration used throughout.

4 Results

4.1 Reinforcement, Not Competition

We use "reinforcement" for three distinct effects, kept separate throughout: *marginal* (joint training raises weight or activation sparsity above what the corresponding solo penalty reaches), *statistical* (a nonzero $\text{Cov}(X, Y)$ in Proposition 4's decomposition – genuine association between which weights and which activations survive, beyond marginal shifting), and *effective-compute* (the aggregate effective-MAC fraction beating the naive independence estimate, to which the first two can each contribute).

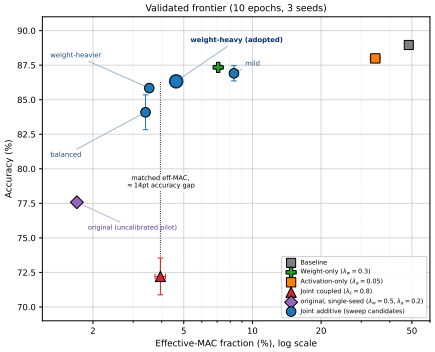


Figure 1: Validated frontier: accuracy vs. effective-MAC fraction, full protocol (10 epochs, 3 seeds; single-seed for the original configuration, shown for reference). See Appendix C for the screening sweep that selects this configuration.

Table 1 reports the five-regime comparison at $n = 5$ seeds per arm, upgraded from $n = 3$ for this headline comparison (every other table remains at $n = 3$ unless noted). If the two penalties competed, joint additive would land between or below the two solo results; instead it exceeds both on the marginal axis: $86.75 \pm 0.10\%$ weight sparsity vs. weight-only's $86.08 \pm 0.43\%$, and $67.72 \pm 2.15\%$ activation sparsity vs. activation-only's $63.08 \pm 1.75\%$. Both gaps are individually significant before correction ($p = 0.033$ and $p = 0.011$); after Benjamini-Hochberg correction across the full family of 12 such tests (Appendix L), the activation-sparsity gap survives ($p_{\text{BH}} = 0.026$) while the weight-sparsity gap falls just short ($p_{\text{BH}} = 0.065$) – 5 of 12 comparisons overall now survive correction, up from 4 of 12 at $n = 3$. The direction is consistent across settings (11 of 12 comparisons reinforcing), though we do not read this as a formal hypothesis test: these comparisons share architecture family, regularizer, and implementation, so they are not independent replications in the sense a sign test assumes. The effective-MAC fraction (4.63%) beats the naive independence estimate ($\approx 5.14\%$) by 9.9% relatively – a reduction reaching 25.8% under an earlier, more aggressive, uncalibrated setting, suggesting reinforcement strength scales with regularization strength at a steep accuracy cost. Figure 1 places all five regimes on this accuracy-vs.-effective-MAC frontier together with the four coefficient-sweep candidates (Appendix C) that weight-heavy was selected from: the sweep spans a range of accuracy/compute trade-offs, all of them well above the single-seed, uncalibrated pilot point ("original") that motivated running the sweep in the first place, and joint coupled reaches a similar effective-MAC fraction to weight-heavy at a 14-point accuracy cost – the matched-compute comparison Section 4.2 investigates next.

Reinforcement is not specific to Hoyer: a naive per-element-normalized L1 penalty at comparably-scaled coefficients reinforces even more strongly (Appendix E) – weight sparsity by $+7.79$ points and activation sparsity by $+12.29$ points, versus Hoyer's $+0.67$ and $+4.64$, its effective-MAC fraction beating the naive estimate by 52% relatively, more than five times Hoyer's reduction, though at a different, less favorable operating point (lower peak weight sparsity even at far larger coefficients).

Table 1: Five-regime comparison on ResNet-18/CIFAR-10, 10 epochs, mean ± std over 5 seeds, sweep-selected configuration ($\lambda_w = 0.3$, $\lambda_a = 0.05$, $\lambda_c = 0.8$).

Regime	Accuracy	Weight Sp.	Act. Sp.	Eff.-MAC
Baseline	$88.96 \pm 0.19\%$	$4.55 \pm 0.11\%$	$48.68 \pm 0.16\%$	$48.47 \pm 0.15\%$
Weight-only	$87.34 \pm 0.13\%$	$86.08 \pm 0.43\%$	$48.97 \pm 0.19\%$	$7.08 \pm 0.23\%$
Activation-only	$87.99 \pm 0.21\%$	$5.70 \pm 0.36\%$	$63.08 \pm 1.75\%$	$34.57 \pm 1.75\%$
Joint additive	$86.33 \pm 0.35\%$	$\mathbf{86.75 \pm 0.10\%}$	$67.72 \pm 2.15\%$	$4.63 \pm 0.12\%$
Joint coupled	$72.21 \pm 1.33\%$	$70.41 \pm 2.04\%$	$\mathbf{84.55 \pm 1.04\%}$	$\mathbf{3.95 \pm 0.21\%}$

Table 2: Five-regime comparison on ResNet-18/CIFAR-10, 50 epochs, mean ± std over 3 seeds (same coefficients as Table 1). Cf. the 10-epoch numbers in Table 1; the additive-coupled accuracy gap shrinks from 14.12 points at 10 epochs to 2.21 here.

Regime	Accuracy	Weight Sparsity	Activation Sparsity	Effective-MAC Fraction
Baseline	$93.00 \pm 0.02\%$	$4.50 \pm 0.07\%$	$56.22 \pm 0.08\%$	$41.41 \pm 0.10\%$
Weight-only ($\lambda_w = 0.3$)	$92.28 \pm 0.08\%$	$93.86 \pm 0.26\%$	$65.60 \pm 0.12\%$	$2.25 \pm 0.14\%$
Activation-only ($\lambda_a = 0.05$)	$92.37 \pm 0.09\%$	$7.05 \pm 0.25\%$	$77.74 \pm 0.26\%$	$20.94 \pm 0.24\%$
Joint additive (weight-heavy)	$91.56 \pm 0.09\%$	$\mathbf{94.07 \pm 0.15\%}$	$82.56 \pm 0.17\%$	$\mathbf{1.18 \pm 0.07\%}$
Joint coupled ($\lambda_c = 0.8$)	$89.35 \pm 0.34\%$	$89.64 \pm 0.35\%$	$80.92 \pm 0.63\%$	$1.85 \pm 0.06\%$

4.2 The Coupled Formulation Underperforms, and Why

Joint coupled costs 14.12 accuracy points relative to joint additive (Table 1), despite reaching a *lower* (better) effective-MAC fraction and *higher* activation sparsity – it trades accuracy for a different sparsity pattern (much lower weight sparsity, much higher activation sparsity), not dominated on every metric. We test three candidate explanations. Calibration asymmetry: recalibrating λ_c to match additive's effective-MAC fraction exactly still leaves a 14-point gap. Pairing imprecision: an exact weight-activation pairing scheme (matching each weight tensor to the activation its output literally feeds into) gives $70.91 \pm 2.27\%$ accuracy, statistically indistinguishable from the approximate scheme's $72.17 \pm 1.59\%$ – ruled out. Slow convergence: doubling the training budget to 20 epochs shrinks the gap from 14.12 to 6.10 points (coupled improves 11.74 points versus additive's 3.61), continuing to 2.21 at 50 epochs (Table 2) – a smooth, monotonically shrinking sequence, plotted in full in Figure 2. We confirm this directly by instrumenting per-step gradient norms: the coupled term's gradient reaching each operand is throttled by the partner operand's current Hoyer value rather than by the operand's own proximity to its target, by a factor of 4.7–$22.9\times$ across the three layers tracked (Appendix G).

Joint coupled recovers to 96.08% of baseline accuracy at 50 epochs, while joint additive is essentially flat (98.45% vs. 97.19% at 10 epochs); the effective-MAC ordering flips too, coupled going from better than additive at 10 epochs to clearly worse at 50 (1.85% vs. 1.18%). Figure 2 plots the full trajectory rather than only these endpoints, for both CIFAR-10 and CIFAR-100. The same convergence-speed pattern replicates on CIFAR-100 ($15.31 \to 5.89$ at 10/50 epochs) and on structured sparsity ($37.10 \to 6.32$; Section D.6), though CIFAR-100 shows a decelerating rather than proportional closure by 100 epochs ($\to 3.47$), and MobileNetV2 closes only partially by 50 epochs ($34.29 \to 17.57$). We conclude the coupled objective's underperformance is substantially a convergence-speed artifact – a product term's gradient on one factor is scaled by the other's current value, plausibly stalling progress once either factor nears zero – rather than a fixed calibration or pairing problem, though a small residual gap and architecture/task interactions (Section 4.3) remain. Used *on top of* the additive objective at a modest coefficient, the coupled term dominates standalone coupled training on every metric and reaches compute reduction pure additive alone cannot (Appendix I); Appendix J reports full tables for every extended-training-budget checkpoint.

This failure turns out to be a fact about Hoyer, not about product-form coupling. Hoyer measures *concentration* (how unevenly mass is distributed across coordinates), not *density* (what fraction of coordinates are effectively zero) – the two coincide at the extremes but differ in between, and the gradient-throttling mechanism above is specifically a property of the coupled *concentration* product. Substituting a density-aligned surrogate, $\hat{\rho}(x) = \mathrm{mean}_i[x_i^2/(x_i^2 + \tau^2)]$ ($\tau = 0.01$), for Hoyer throughout – including in the coupled product $\hat{\rho}_W(W_l) \cdot \hat{\rho}_A(A_l)$ – changes the picture (Table 3): at the identical n=3 protocol, Hoyer's coupled objective costs additive 14.23 points ($p < 0.0001$), while the density surrogate's gap is just 0.30 points, statistically indistinguishable from additive ($p = 0.89$) – not a smaller version of the same failure, but no detectable accuracy failure at all. The lopsided sparsity allocation persists regardless (weight sparsity 57.12% vs. 81.65%; activation

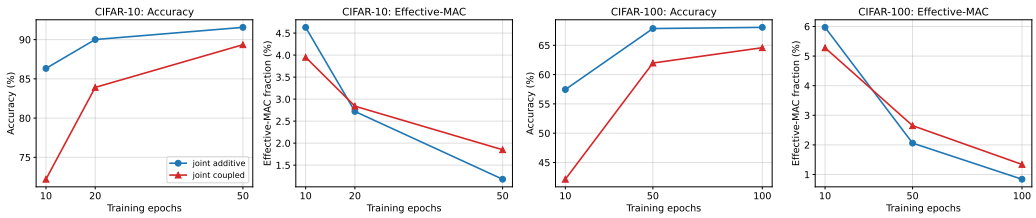


Figure 2: Joint additive vs. joint coupled accuracy and effective-MAC fraction across every training-budget checkpoint tested, ResNet-18/CIFAR-10 (left pair, 10/20/50 epochs) and ResNet-18/CIFAR-100 (right pair, 10/50/100 epochs), mean over 3 seeds (error bars omitted for legibility; see Appendix J for standard deviations). Both effective-MAC panels show the crossover described in the text: coupled starts with a lower (better) fraction at 10 epochs and additive overtakes it by the final checkpoint.

Table 3: Five-regime comparison using the density surrogate $\hat{\rho}$ in place of Hoyer, ResNet-18/CIFAR-10, 10 epochs, mean $\pm$ std over 3 seeds. Cf. Table 1's Hoyer numbers at the identical protocol.

Regime	Accuracy	Weight Sparsity	Activation Sparsity	Effective-MAC Fraction
Baseline	$88.83 \pm 0.22\%$	$4.65 \pm 0.14\%$	$48.70 \pm 0.10\%$	$48.35 \pm 0.07\%$
Weight-only ($\lambda_w = 0.2$)	$84.47 \pm 0.61\%$	$80.59 \pm 0.56\%$	$49.99 \pm 0.18\%$	$9.63 \pm 0.24\%$
Activation-only ($\lambda_a = 0.05$)	$88.16 \pm 1.11\%$	$5.52 \pm 0.60\%$	$68.97 \pm 2.68\%$	$29.18 \pm 2.80\%$
Joint additive	$83.66 \pm 1.34\%$	$\mathbf{81.65 \pm 0.94\%}$	$72.61 \pm 2.58\%$	$\mathbf{5.58 \pm 0.51\%}$
Joint coupled ($\lambda_c = 0.2$)	$83.36 \pm 2.55\%$	$57.12 \pm 0.61\%$	$81.31 \pm 1.63\%$	$6.50 \pm 0.79\%$

sparsity 81.31% vs. 72.61%), so that signature is not Hoyer-specific, but the severe *accuracy* collapse studied throughout this paper is (Appendix R discusses the mechanism further).

Table 3 also shows the density surrogate is not uniformly gentler than Hoyer: weight-only's own accuracy cost roughly triples under $\hat{\rho}$ (84.47% vs. Hoyer's 87.34% at a comparably-scaled coefficient, Table 1), even though joint additive and joint coupled both end up close to Hoyer's joint-additive accuracy. Activation-only is largely unaffected (88.16%, matching Hoyer's 87.99%). So the surrogate's rescue of the coupled objective is not explained by $\hat{\rho}$ being a globally milder penalty – it specifically leaves the coupled *interaction*, not each solo regime, in a better place.

4.3 Generalization: Architecture, Task Difficulty, Domain, and Granularity

We test six further settings, summarized in Table 4 (full per-setting tables in Appendix D): task difficulty alone (ResNet-18/CIFAR-100, independently-swept coefficients), architecture and task jointly (LeNet/MNIST, un-tuned CIFAR-10 coefficients), domain shift alone (ResNet-18/SVHN, class count and architecture fixed), architecture alone (MobileNetV2/CIFAR-10), architecture $\times$ task difficulty jointly (MobileNetV2/CIFAR-100), and sparsity granularity (structured, channel-level Hoyer on ResNet-18/CIFAR-10 and CIFAR-100).

Reinforcement is real but conditional, not uniform. It replicates more clearly on CIFAR-100 than CIFAR-10 on both architectures tested, and transfers cleanly to SVHN, this paper's best accuracy-sparsity trade-off (joint additive costs only 0.27 accuracy points for $92.5\%/80.2\%$ sparsity). On MobileNetV2/CIFAR-10 both marginal sparsities still reinforce, but the aggregate effective-MAC fraction does *not* beat the naive estimate (10.30% actual vs. 9.74% naive) – the only setting where it fails to, though the harder task restores it (MobileNetV2/CIFAR-100: 10.02% vs. 10.84% naive). On MNIST, activation-sparsity reinforcement reverses sign, and joint coupled reaches the *highest* accuracy of any regularized regime – among vision settings, the only place the coupled formulation wins, plausibly because MNIST/LeNet varies architecture, task, and calibration simultaneously relative to the sweep-selected configuration.

Structured (channel-level) sparsity makes reinforcement an order of magnitude larger and reliably significant. Under structured sparsity (Table 6), reinforcement clears significance decisively ($p = 0.0006$ and $p = 0.0067$ on CIFAR-10), unlike every unstructured comparison. This is a deterministic strengthening of Proposition 2's mechanism, not a new one: element-wise weight sparsity only *shrinks* a unit's pre-activation variance, making a zero more likely but never certain, whereas a fully-zeroed output channel's pre-activation is exactly zero at every spatial position of every example – the entire downstream activation map is guaranteed zero rather than merely encouraged toward it, a plausible reason the effect is reliably significant where the unstructured, probabilistic version is not.

Table 4: Generalization summary: joint additive vs. its two solo regimes, and the joint additive–coupled accuracy gap, across all settings tested. "Sig." marks whether the joint-additive-vs-joint-coupled accuracy gap survives Benjamini-Hochberg correction at $\alpha = 0.05$ (n=5 seeds for ResNet-18/CIFAR-10, n=3 elsewhere; Appendix L). SVHN's coupled regime is bimodal (Section 4.3) and excluded from the gap column.

Setting	Joint add. acc.	Weight sp. gap	Act. sp. gap	Add.-coupled gap	Sig.
ResNet-18/CIFAR-10 (Table 1, n=5)	86.33%	+0.67	+4.64	14.12 pts	Yes
ResNet-18/CIFAR-100	57.45%	+1.37	+5.95	15.31 pts	Yes
LeNet/MNIST	97.32%	+9.68	−2.91	coupled *wins*	Yes
ResNet-18/SVHN	96.15%	+0.44	+6.70	bimodal (n/a)	–
MobileNetV2/CIFAR-10	80.32%	+1.30	+0.39	34.29 pts	Yes
MobileNetV2/CIFAR-100	46.22%	+2.58	+3.20	35.26 pts	Yes
Structured, CIFAR-10	86.66%	+11.50	+13.25	37.10 pts	–
Structured, CIFAR-100	58.24%	+8.16	+13.34	36.50 pts	–

Table 5: Five-regime comparison on a word-level Transformer LM, WikiText-2, 3000 steps, mean ± std over 3 seeds. Lower validation loss and perplexity are better (unlike every accuracy-based table elsewhere in this paper).

Regime	Val. Loss	Perplexity	Weight Sp.	Act. Sp.	Eff.-MAC
Baseline	5.6046 ± 0.0057	271.67 ± 1.55	$2.40 \pm 0.05\%$	$50.45 \pm 0.39\%$	$48.40 \pm 0.34\%$
Weight-only ($\lambda_w = 0.1$)	5.7326 ± 0.0050	308.77 ± 1.55	$66.51 \pm 0.09\%$	$48.82 \pm 0.65\%$	$17.09 \pm 0.20\%$
Activation-only ($\lambda_a = 0.05$)	5.6335 ± 0.0109	279.65 ± 3.05	$2.77 \pm 0.02\%$	$77.26 \pm 0.16\%$	$22.69 \pm 0.15\%$
Joint additive	5.7510 ± 0.0044	314.51 ± 1.38	$\mathbf{68.94 \pm 0.32}\%$	$76.63 \pm 0.73\%$	$\mathbf{8.48 \pm 0.14}\%$
Joint coupled ($\lambda_c = 0.1$)	5.7235 ± 0.0059	305.97 ± 1.81	$4.30 \pm 0.02\%$	$\mathbf{90.45 \pm 0.95}\%$	$9.24 \pm 0.94\%$

A second, larger such win appears outside vision entirely, and part of it is now validated rather than merely preliminary. We test a small causal Transformer LM on two corpora: character-level Tiny Shakespeare (single seed; Appendix O) and, at n=3 seeds, the standard word-level WikiText-2 benchmark, 19× larger (Table 5; Appendix P). Both show the same marginal-reinforcement split (weight sparsity reinforces, activation sparsity reverses) also seen in a companion ReLU-ViT probe, and both show joint coupled beating joint additive on validation loss – significantly so on WikiText-2 ($p = 0.0038$), the opposite of every unstructured vision result. The character-level probe's more dramatic claim, that coupled beats even the unregularized baseline and reaches the lowest effective-MAC fraction of any regime, does not repeat at word-level scale, where coupled beats additive only. We read the validated result as the most direct challenge in this paper to its own headline finding.

The same question, asked with a completely different weight-sparsity mechanism on real deployed hardware-target networks, still finds conditional rather than uniform reinforcement. Appendix S repeats this paper's core question – does jointly pushing activation and weight sparsity reinforce or compete? – on four production edge-AI architectures (Visual Wake Words, PlantVillage, Speech Commands, ECG arrhythmia classification), replacing the differentiable Hoyer weight penalty used everywhere else in this paper with hard, discrete Network Slimming channel pruning (Liu et al., 2017). All four patterns seen elsewhere in this paper recur, but distributed one-per-model rather than clustered: complementary throughout (VWW), antagonistic by an order of magnitude at aggressive settings yet resolved by scaling down (PlantVillage), a genuine sign flip between operating points within one model (Speech Commands), and outright accuracy-*improving* joint sparsification (ECG arrhythmia). No two of the four agree, independent evidence for this paper's central claim from a different weight-sparsity mechanism entirely.

The coupled objective's disadvantage compounds with task difficulty rather than adding independently. The additive-coupled accuracy gap widens by a similar ~ 1 point on both architectures when task difficulty alone increases (ResNet-18: $14.12 \rightarrow 15.31$; MobileNetV2: $34.29 \rightarrow 35.26$), looking additive; but relative accuracy retention under coupled training drops 13.3 points on ResNet-18 (CIFAR-10→CIFAR-100) versus 33.7 points on MobileNetV2 – MobileNetV2/CIFAR-100 collapses to 20.9% relative retention ($10.96 \pm 6.64\%$ accuracy, one seed at chance level), a high-variance, bimodal-flavored failure distinct from SVHN's but similarly severe, at a coefficient recalibrated for the harder *task* but not for this *architecture*. Recalibrating λ_c specifically for MobileNetV2 ($\lambda_c = 0.23$, screened to match joint additive's own effective-MAC target) substantially improves both accuracy and stability: $33.06 \pm 1.49\%$ vs. $10.96 \pm 6.64\%$, seed variance shrinking more than fourfold (Appendix D.5). This confirms the calibration-asymmetry mechanism concretely: the extreme failure was not a fundamentally uncontrollable instability the way SVHN's is, but a genuine architecture-driven miscalibration that an architecture-specific coefficient substantially, though not fully, resolves – a 33.4-point gap to joint additive's 66.49% remains.

Table 6: Five-regime comparison with *structured* (channel-level) weight sparsity on ResNet-18/CIFAR-10, 10 epochs, mean ± std over 3 seeds, reusing the sweep-selected coefficients unchanged (Sec 4.3). Weight sparsity here is the fraction of near-zero-norm output channels, not individual weights.

Regime	Accuracy	Channel Sparsity	Activation Sparsity	Effective-MAC Fraction
Baseline	$88.67 \pm 0.18\%$	$0.00 \pm 0.00\%$	$48.65 \pm 0.20\%$	$50.76 \pm 0.26\%$
Weight-only ($\lambda_w = 0.3$)	$88.47 \pm 0.02\%$	$48.82 \pm 0.40\%$	$54.68 \pm 0.54\%$	$23.68 \pm 0.16\%$
Activation-only ($\lambda_a = 0.05$)	$87.94 \pm 0.42\%$	$0.00 \pm 0.00\%$	$63.09 \pm 1.93\%$	$36.63 \pm 2.02\%$
Joint additive (weight-heavy)	$86.66 \pm 0.65\%$	$\mathbf{60.32 \pm 0.03\%}$	$\mathbf{76.35 \pm 0.57\%}$	$\mathbf{10.81 \pm 0.10\%}$
Joint coupled ($\lambda_c = 0.8$, unrecalibrated)	$49.56 \pm 10.94\%$	$1.62 \pm 0.88\%$	$93.99 \pm 1.41\%$	$5.24 \pm 1.20\%$

SVHN's coupled regime is a qualitatively different phenomenon. Across ten seeds, two converge normally ($\approx 94.6\%$) and eight collapse to an identical 19.59–19.64% (SVHN's majority-class frequency), persisting across a λ_c sweep from 0.2 to 0.8 with no monotonic trend, invisible in initialization statistics and early-training accuracy, and not even a deterministic function of the nominal seed – repeated runs of the same seed land in different basins, pointing to execution-level floating-point nondeterminism rather than ordinary seed-dependent bistability. We confirm this directly: forcing bit-deterministic execution makes both previously-flip-flopping seeds perfectly reproducible across three reruns each (Appendix D.3).

4.4 Effective-MAC Validation

We instrument the real forward pass (im2col plus a boolean-mask matrix multiply, no independence assumption) to compute the exact executed-MAC fraction and compare it against the proxy. The two diverge for the dense baseline (48.15% proxy vs. 41.02% exact, MAC-weighted) but converge closely for every regularized regime, within 0.01–0.64 points for joint coupled and joint additive (Appendix K) – reassuring for this paper's central compute-savings comparisons even though the independence assumption is measurably imperfect in general. The measured covariance term (Proposition 4) is small but reliably positive and an order of magnitude larger under joint regularization ($+0.67$pp) than under either solo penalty ($+0.07$–0.08pp), direct evidence that joint training induces genuine statistical association between which weights and which activations survive, not merely two independent marginal reductions.

5 Discussion, Limitations, and Future Work

Interventions against the coupled objective's convergence-speed mechanism. We test two mitigations targeted directly at the throttling mechanism Section 4.2 identifies (Appendix H): a log-space reformulation ($\lambda_c \sum_l [\log H(W_l) + \log H(A_l)]$, removing each factor's gradient scaling by the other's magnitude) and gradient clipping ($\|\cdot\| \leq 1$) on standard product-coupled training. Neither closes the gap to additive, but each moves the coupled objective to a different, informative point: log-space reaches a substantially lower effective-MAC fraction ($2.70 \pm 0.22\%$ vs. $3.93 \pm 0.27\%$, a 31% relative reduction) at statistically indistinguishable accuracy ($70.61 \pm 3.22\%$ vs. $72.17 \pm 1.59\%$), while gradient clipping raises accuracy by 3.02 points ($75.19 \pm 1.84\%$, closing roughly a fifth of the gap to additive) at a small effective-MAC cost – evidence the optimization-difficulty mechanism is real and partially addressable, though neither is a full fix. A separate, direct causal test of Proposition 2's mechanism – permanently freezing a realistic weight-sparsity structure *before* activation regularization begins, rather than letting both develop together – does not reproduce reinforcement; if anything it runs backward, with an unfrozen continuation reaching higher accuracy *and* higher activation sparsity than either a realistic or a shuffled frozen mask (Appendix Q). We read this as evidence the mechanism depends on joint training's continuous, simultaneous pressure rather than surviving translation into a fixed prior constraint – not as evidence against the correlational reinforcement findings established throughout this paper.

SVHN's instability is a distinct phenomenon, not the same mechanism at a more extreme setting. Gradient clipping does not touch it: across ten seeds at the identical clip norm, $8/10$ still collapse, matching the unclipped rate – evidence against gradient scale as the driver. Two further interventions, learning-rate warmup and an SGD-with-momentum optimizer in place of Adam, both leave $10/10$ seeds collapsed, at or above the untreated baseline's $8/10$ (Appendix D.3). The instability is coefficient-independent, state-independent, and only partially seed-dependent, closer to execution-sensitive basin selection than to ordinary instability: forcing bit-deterministic execution makes both previously-flip-flopping seeds perfectly reproducible across three reruns each.

Scope and statistical power. The headline ResNet-18/CIFAR-10 result is a single-architecture proof of concept, though no longer single-seed, and we address transfer across three independent axes (task difficulty, domain, architecture) plus their pairwise interaction and one granularity axis (structured vs. unstructured). Every fully-validated architecture, however, is a convolutional network on a vision benchmark, with two exceptions: a non-CNN vision architecture (ViT, still single-seed and preliminary) and a non-CNN, non-vision one – a causal Transformer LM, now validated at n=3 seeds on a standard word-level benchmark (WikiText-2, Appendix P), though still only one architecture/corpus pair with screened-not-swept coefficients. Reinforcement's individual marginal-sparsity comparisons are directionally consistent (12/14 positive) but mostly not individually significant at $n = 3$ seeds (Appendix L) – a genuine power limitation, not a technicality. The four production edge-AI architectures (Appendix S) extend scope along architecture, task domain, and weight-sparsity mechanism simultaneously, but each grid point there is a single run rather than a seed-averaged one, and the design is a handful of hand-chosen (λ, p) pairs rather than a coefficient sweep – appropriate for the engineering pilots they originally were, but weaker evidence per data point than this paper's main comparisons. Remaining open directions include training CIFAR-100 well beyond 100 epochs to see whether its coupled gap asymptotes above zero, recalibrating λ_c for the structured-sparsity case, and finding an intervention – clipping, warmup, and an alternative optimizer having each now been ruled out – against SVHN's instability.

Reinforcement at matched coefficients is not the same claim as an accuracy advantage at matched compute, and the paper's own evidence does not support the latter. Every reinforcement result above holds the regularization *coefficient* fixed across regimes. A practitioner choosing between methods at a fixed *compute* target faces a different question: pushing a solo weight-only penalty to reach joint additive's own effective-MAC fraction reaches statistically indistinguishable accuracy ($86.63 \pm 0.26\%$ vs. $86.33 \pm 0.35\%$, $p = 0.21$, $n = 5$), with the point estimate favoring the *solo* penalty (Appendix F). Reinforcement changes *how* a given compute reduction is achieved – a more balanced weight/activation split versus a solo penalty's near-total reliance on one operand – without demonstrating that it lowers *what that reduction costs* in accuracy.

6 Conclusion

Jointly regularizing weight and activation sparsity with a simple additive objective reinforces both, in most but not all settings we test, more strongly and reliably under structured (channel-level) sparsity than under the unstructured case most prior work studies. This changes *how* a given compute reduction is achieved, not *what it costs*: at matched effective-MAC fraction, a solo weight-only penalty reaches statistically indistinguishable accuracy to joint additive (Section 5), so we do not claim joint training improves the accuracy-compute frontier itself, only that it reaches a given point through a more balanced weight/activation split. A theoretically better-motivated coupled (product) objective instead underperforms substantially under Hoyer – largely a slow-convergence, gradient-throttling artifact confirmed directly by instrumentation, not a fixed calibration or pairing problem. This failure is specific to Hoyer, not product-form coupling in general: a density-aligned surrogate in the identical objective closes a 14.23-point accuracy gap to 0.30 points, statistically indistinguishable from additive, while the underlying lopsided sparsity allocation persists – so the coupled formulation's appeal need not be abandoned along with Hoyer.

Generalization is real but conditional: it depends on architecture, task difficulty, domain, and sparsity granularity, sometimes compounding destructively (MobileNetV2 $\times$ CIFAR-100), sometimes producing qualitatively different failure modes (SVHN's bimodal instability), and, outside vision entirely, sometimes reversing outright: the coupled objective beats additive on both a preliminary character-level probe and a validated, n=3 word-level benchmark, though only the smaller corpus also beats an unregularized baseline. This conditionality is not an artifact of studying one weight-sparsity mechanism on research benchmarks: repeating the same question on four independent production edge-AI architectures with hard Network Slimming pruning in place of Hoyer (Appendix S) recovers all four patterns seen elsewhere in this paper – complementary, antagonistic, sign-flipping, and accuracy-improving – one per model, with no two agreeing. We view weight and activation sparsity as jointly, rather than independently, optimized properties of the effective computation graph, with SVHN's instability, the coupled-Hoyer mechanism's remaining gap, and non-vision generalization as concrete open questions.

Reproducibility Statement

All experiments use a single, shared training pipeline (Adam, cosine annealing, standard augmentation) described in Section 3.3; every three-seed table reports mean $\pm$ std over seeds $\{0, 1, 2\}$ with no seed selection, and Table 1 (upgraded to $n = 5$ for the headline comparison, Section 4.1) extends this to seeds $\{0, 1, 2, 3, 4\}$. Proofs of all four propositions are complete and self-contained (Appendix A). The effective-MAC proxy is validated against an independent, assumption-free exact instrumentation method (Section 4.4, Appendix K), and every statistical claim of significance is reported with the underlying test, n, and multiple-comparison correction (Appendix L). SVHN's bimodal instability (Section 4.3) is itself a reproducibility finding – we report the seed-level nondeterminism directly, including that repeated runs of an identical seed land in different outcomes, rather than only a mean and standard deviation that would misrepresent it.

AI Use Statement

Generative AI (an LLM-based coding assistant) was used to help implement experiment code, run and monitor training jobs, generate figures, and draft and revise text throughout this paper, under continuous author direction and review. All experimental design decisions, choice of what to report, and interpretation of results were made and verified by the authors; all numerical results were produced by code the authors reviewed and re-ran to confirm, not accepted from AI output without verification. We have not used generative AI to generate synthetic experimental data or fabricate results. We take full responsibility for the final content of this work, including all text, claims, and code.

References

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

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

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

Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. Eie: Efficient inference engine on compressed deep neural network. In *International Symposium on Computer Architecture (ISCA)*, 2016.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *International Conference on Computer Vision (ICCV)*, 2017.

Patrik O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

Mark Kurtz, Justin Kopinsky, Rati Gelashvili, Alexander Matveev, John Carr, Michael Goin, William Leiserson, Sage Moore, Bill Nell, Nir Shavit, and Dan Alistarh. Inducing and exploiting activation sparsity for fast neural network inference. In *International Conference on Machine Learning (ICML)*, 2020.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations (ICLR)*, 2017.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *International Conference on Computer Vision (ICCV)*, 2017.

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

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*, 2017.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

Md Aamir Raihan and Tor M. Aamodt. Sparse weight activation training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Huanrui Yang, Wei Wen, and Hai Li. Deephoyer: Learning sparser neural network with differentiable scale-invariant sparsity measures. In *International Conference on Learning Representations (ICLR)*, 2020.

Qing Yang, Wei Wen, Zuoguan Wang, and Hai Li. Joint regularization on activations and weights for efficient neural network pruning. *arXiv preprint arXiv:1906.07875*, 2019.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. Learning n:m fine-grained structured sparse neural networks from scratch. In *International Conference on Learning Representations (ICLR)*, 2021.

Zeqi Zhu, Arash Pourtaherian, Luc Waeijen, Lennart Bamberg, Egor Bondarev, and Orlando Moreira. Arts: An adaptive regularization training schedule for activation sparsity exploration. In *Euromicro Conference on Digital System Design (DSD)*, 2022.

A PROOFS

Proposition 1 (gradient shielding). *Let $z = \sum_i w_i a_i$ be a unit's pre-activation and $y = \mathrm{ReLU}(z)$ its post-activation, feeding a downstream loss L. For any training example on which $a_i = 0$, $\partial L/\partial w_i = 0$ exactly. Proof.* By the chain rule, $\partial L/\partial w_i = (\partial L/\partial z) \cdot (\partial z/\partial w_i) = (\partial L/\partial z) \cdot a_i$, which is zero whenever $a_i = 0$ regardless of $\partial L/\partial z$. ■ This is exact but per-example and per-step; it says nothing about w_i's trajectory over the many steps on which $a_i \neq 0$.

Proposition 2 (variance shrinkage). *Let $a_1, \ldots, a_n$ be independent with $\mathrm{Var}(a_i) = \sigma_i^2 < \infty$, and $z = \sum_i w_i a_i$. Then $\mathrm{Var}(z) = \sum_i w_i^2 \sigma_i^2$, which weakly decreases as any $|w_i| \to 0$, strictly so whenever $\sigma_i > 0$. Proof.* Immediate from the variance of a sum of independent random variables, $\mathrm{Var}(\sum_i w_i a_i) = \sum_i w_i^2 \mathrm{Var}(a_i)$, since $\mathrm{Cov}(a_i, a_j) = 0$ for $i \neq j$. ■ A smaller-variance pre-activation is not by itself proof ReLU is more likely to zero it exactly (that also depends on the mean and distributional shape), but gives a precise, quantitative handle on the intuition.

Proposition 3 (range of the raw squared-Hoyer ratio). *For any $x \in \mathbb{R}^N \setminus \{0\}$, define $\rho(x) = (\sum_i |x_i|)^2 / \sum_i x_i^2$. Then $1 \leq \rho(x) \leq N$. The lower bound is tight iff at most one coordinate of x is nonzero; the upper bound is tight iff all nonzero coordinates have equal magnitude. Proof.* Expanding the square, $(\sum_i |x_i|)^2 = \sum_i x_i^2 + 2\sum_{i<j} |x_i||x_j| \geq \sum_i x_i^2$, since every cross term is non-negative; dividing by $\sum_i x_i^2$ gives $\rho(x) \geq 1$, with equality iff every cross term vanishes, i.e. at most one $|x_i|$ is nonzero. For the upper bound, Cauchy-Schwarz applied to $(|x_1|, \ldots, |x_N|)$ and $(1, \ldots, 1)$ gives $(\sum_i |x_i| \cdot 1)^2 \leq (\sum_i x_i^2)(\sum_i 1^2) = N \sum_i x_i^2$, so $\rho(x) \leq N$, with equality iff $|x_i|/1$ is constant across all i. ■ The practical content: $\rho(x)$'s achievable range grows linearly with N, so a fixed coefficient λ exerts pressure whose maximum magnitude differs by $N_{\text{large}}/N_{\text{small}}$ between layers, regardless of how sparse either actually is. Dividing by N collapses this to $[1/N, 1]$, well-approximated by $(0, 1]$ for any layer of practical size.

Proposition 4 (exact decomposition of the effective-MAC fraction). *Let X, Y be the $\{0,1\}$-valued indicator variables for "weight operand nonzero" and "activation operand nonzero", respectively, over the empirical distribution of (weight, activation) operand pairs that layer l's real convolution sums range over. Then $C_l^{\text{exact}} = P(X{=}1, Y{=}1)$ satisfies $C_l^{\text{exact}} = P(X{=}1)P(Y{=}1) + \mathrm{Cov}(X, Y)$. Proof.* Immediate from $\mathrm{Cov}(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$ together with $\mathbb{E}[XY] = P(X{=}1, Y{=}1)$ for $\{0,1\}$-valued X, Y. ■ This separates two effects reinforcement claims otherwise conflate: joint training raising the marginals $P(X{=}1)$, $P(Y{=}1)$ changes C_l through the independence term alone; only a nonzero covariance reflects genuine within-layer statistical association between which weights and which activations are zero.

B A MINIMAL TOY EXAMPLE ILLUSTRATING THE MECHANISM

Propositions 1 and 2 identify mechanisms consistent with reinforcement but, as stated above, do not by themselves establish that joint training reaches a sparser state than either solo penalty. We make the mechanism concrete with the smallest system that contains it: a single two-operand product $z = wa$ trained by proximal (sub)gradient descent under an L_1 penalty – the same finite-time-exact-zeroing behavior Hoyer regularization also relies on (Proposition 3's ratio is sparsest when one coordinate absorbs all the mass, and coordinate-wise proximal L_1 steps reach exact zero in finite time whenever the local subgradient budget exceeds the opposing loss gradient, unlike a smooth penalty such as ridge, which only asymptotes to zero). We consider two cases isolating the two directions Proposition 1's shielding argument can cut.

Case A: a redundant unit. Let $w, a \in \mathbb{R}$ with loss $L = \frac{1}{2}(wa)^2 + \lambda_w|w| + \lambda_a|a|$ – a unit the task does not need, so driving either operand to zero costs nothing. Proximal gradient descent with step size η updates $w \leftarrow S_{\eta\lambda_w}(w - \eta w a^2)$, $a \leftarrow S_{\eta\lambda_a}(a - \eta a w^2)$, where $S_t(x) = \mathrm{sign}(x)\max(|x| - t, 0)$ is the soft-threshold operator. From $w_0 = a_0 = 1$, $\eta = 0.05$, $\lambda = 0.05$ on whichever operand(s) are regularized: weight-only ($\lambda_w = \lambda$, $\lambda_a = 0$) drives w to exact 0 at step 129, but a – receiving no direct penalty, and whose task-gradient aw^2 vanishes identically once $w = 0$ – freezes at $a = 0.419$ *forever* (verified to 10^5 steps); activation-only is the mirror image. Joint additive ($\lambda_w = \lambda_a = \lambda$) reaches *exact* $w = a = 0$ at step 167 – 38 steps later than weight-only's own w, but achieving what neither solo run ever does: full joint sparsity. This is Proposition 1 exactly: once a hits its own regularizer's hard zero, w's task-gradient vanishes identically, so w's remaining descent to zero costs the loss

Table 7: Joint-additive candidates from the coefficient sweep, full protocol (10 epochs, mean ± std over 3 seeds). **weight-heavy** is adopted as the primary configuration (Table 1).

Candidate	Accuracy	Weight Sparsity	Activation Sparsity	Effective-MAC Fraction
mild ($\lambda_w = 0.1, \lambda_a = 0.05$)	$86.91 \pm 0.55\%$	$77.56 \pm 0.51\%$	$65.20 \pm 2.33\%$	$8.30 \pm 0.34\%$
weight-heavy ($\lambda_w = 0.3, \lambda_a = 0.05$)	$86.40 \pm 0.40\%$	$86.75 \pm 0.13\%$	$67.57 \pm 2.38\%$	$4.64 \pm 0.12\%$
weight-heavier ($\lambda_w = 0.5, \lambda_a = 0.05$)	$85.84 \pm 0.07\%$	$89.77 \pm 0.07\%$	$68.36 \pm 2.42\%$	$3.53 \pm 0.09\%$
balanced ($\lambda_w = 0.3, \lambda_a = 0.1$)	$84.09 \pm 1.26\%$	$87.59 \pm 0.19\%$	$76.32 \pm 1.53\%$	$3.40 \pm 0.14\%$

nothing and is driven purely by λ_w – symmetrically once w zeroes first. A solo penalty leaves the *unpenalized* operand permanently stuck away from zero; only joint training finishes both.

Case B: a task-relevant unit. Now let the task require $wa \approx c$ for $c > 0$: $L = \frac{1}{2}(wa - c)^2 + \lambda_w|w| + \lambda_a|a|$. From the same initialization and coefficient ($c = 1$, $\lambda = 0.05$, 2000 steps), weight-only reaches $w = 0.407$ by letting a grow unopposed to 2.405 to preserve the product; joint additive, denied that escape valve on either side, settles at the symmetric point $w = a = 0.974$ – *less* sparse on w than weight-only alone. Regularizing only one operand of a genuinely useful unit lets the other compensate, manufacturing sparsity in the regularized operand at the other's expense; regularizing both removes the compensation channel and can suppress marginal sparsity relative to a solo penalty.

These two cases are two ends of the same mechanism, not two different ones, and neither is more "correct" than the other – each is simply what the shared shielding mechanism does under a different condition on the unit it acts on. On a unit whose task-optimal state is zero for both operands (Case A), shielding lets joint regularization finish sparsifying both, which a solo penalty structurally cannot do alone (reinforcement). On a unit the task genuinely needs (Case B), the identical mechanism denies the compensation escape a solo penalty would otherwise allow, spreading the required non-zero product across both operands instead of concentrating the cost in the unregularized one (suppression) – and this is arguably the *desired* outcome for a genuinely useful unit, since it keeps both operands moderately sized rather than letting the free one grow without bound to protect the task's output. A real network mixes both regimes across its many weight–activation pairs – most prunable at any nonzero regularization, a few carrying the signal keeping accuracy nonzero – which is why the empirical outcome is neither uniform reinforcement nor uniform competition, but the setting-dependent mixture Section 4.3 in fact reports (reinforcing on most marginal-sparsity comparisons, weaker under milder calibration, reversed in aggregate on MobileNetV2/CIFAR-10).

C Coefficient Sensitivity Sweep

The initial calibration ($\lambda_w = 0.5$, $\lambda_a = 0.2$, $\lambda_c = 0.5$, chosen only to avoid collapse) cost nearly 11 accuracy points. We answer whether a milder configuration preserves most of the sparsity at much lower accuracy cost with a two-stage sweep.

Screening. We train every configuration for 5 epochs, single seed: a 4×4 grid over $\lambda_w \in \{0.1, 0.2, 0.3, 0.5\}$ and $\lambda_a \in \{0.05, 0.1, 0.15, 0.2\}$ for joint additive (16 configurations), and $\lambda_c \in \{0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 1.0\}$ for joint coupled (7 configurations), against a matching 5-epoch baseline (83.33% accuracy). Figure 3 shows the resulting trade-off: the joint additive frontier dominates joint coupled's at essentially every effective-MAC level tested.

Validation. Candidate selection is Pareto dominance: among the 16 screened joint-additive configurations, 7 are non-dominated. We carry forward the four least accuracy-costly – *mild*, *weight-heavy*, *weight-heavier*, *balanced* – for full-protocol validation (10 epochs, three seeds; Table 7).

We adopt **weight-heavy** using a quantitative elbow criterion (marginal accuracy cost per point of additional effective-MAC reduction between consecutive frontier candidates): from *mild* to *weight-heavy* this cost is 0.14 points per point of reduction; from *weight-heavy* to *weight-heavier* it jumps $3.6\times$ to 0.50; to *balanced* it jumps a further $26.7\times$ to 13.46. weight-heavy is the last candidate before this cost increases sharply. For joint coupled, we search for the λ_c whose effective-MAC fraction most closely matches weight-heavy's full-protocol target (4.64%) rather than reusing the arbitrarily-calibrated $\lambda_c = 0.5$: a short fine sweep ($\lambda_c \in \{0.6, 0.7, 0.8, 0.9\}$, 5 epochs, single seed) finds $\lambda_c = 0.8$ closest (4.85% at 5 epochs). This proxy match is approximate (it compares a cheap 5-epoch coupled number against weight-heavy's 10-epoch, 3-seed target), and the actual full-protocol

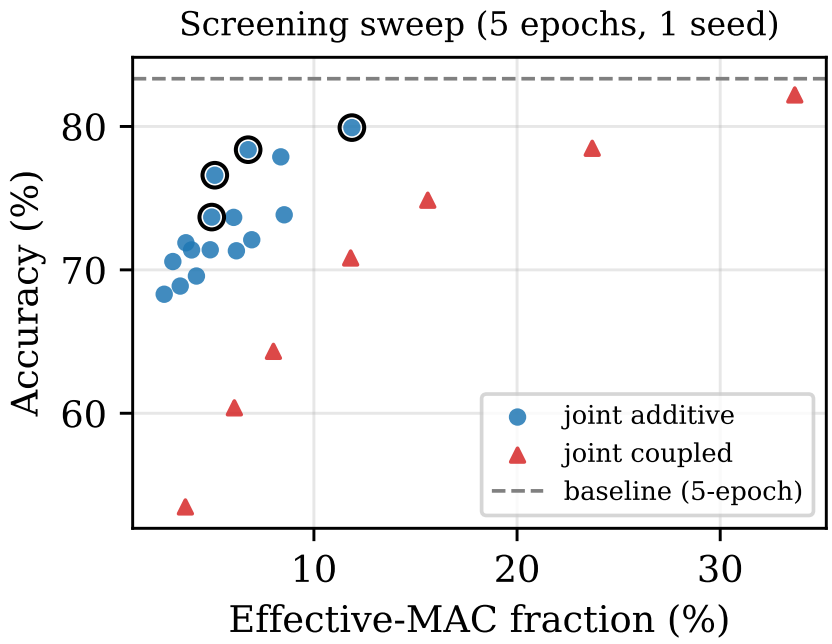


Figure 3: Screening sweep: accuracy vs. effective-MAC fraction for 16 joint-additive and 7 joint-coupled configurations (5 epochs, single seed). Circled points were selected for full-protocol validation.

Table 8: Five-regime comparison on LeNet/MNIST, 8 epochs, mean ± std over 3 seeds. Coefficients unchanged from the ResNet-18/CIFAR-10 initial calibration (Section 3.1), not the sweep-selected one.

Regime	Accuracy	Weight Sparsity	Activation Sparsity	Effective-MAC Fraction
Baseline	$99.12 \pm 0.02\%$	$2.95 \pm 0.07\%$	$53.10 \pm 0.79\%$	$44.10 \pm 2.30\%$
Weight-only ($\lambda_w = 0.5$)	$98.57 \pm 0.12\%$	$74.15 \pm 0.73\%$	$39.63 \pm 3.62\%$	$15.56 \pm 1.10\%$
Activation-only ($\lambda_a = 0.2$)	$98.98 \pm 0.03\%$	$4.12 \pm 0.10\%$	$77.04 \pm 0.87\%$	$20.94 \pm 0.63\%$
Joint additive	$98.49 \pm 0.02\%$	$\mathbf{83.83 \pm 1.30\%}$	$74.13 \pm 1.51\%$	$\mathbf{4.00 \pm 0.31\%}$
Joint coupled ($\lambda_c = 0.5$)	$98.97 \pm 0.07\%$	$59.59 \pm 1.91\%$	$75.93 \pm 1.60\%$	$8.11 \pm 0.81\%$

effective-MAC this produces is 3.93%, not the intended 4.64% (Section 4.2 discusses the direction of this mismatch). Figure 1 (main text) shows the resulting validated frontier.

D Full Generalization Results

This appendix reports the full per-setting tables summarized in Table 4 (Section 4.3).

D.1 LeNet/MNIST

We repeat all five regimes on a LeNet-5-style network (two convolutional layers, three fully-connected layers, four hooked activations, five prunable weight tensors including FC layers) on MNIST. This experiment predates the coefficient sweep (Appendix C): it reuses the *original* $\lambda_w = 0.5$, $\lambda_a = 0.2$, $\lambda_c = 0.5$ calibration without any re-tuning. Coefficients calibrated on one architecture or dataset do not always transfer to another (Section 4.3), but these values remained well-behaved on MNIST/LeNet with no collapse at any regime, across 8 epochs and three seeds. Table 8 reports the results.

Joint additive's weight sparsity (83.83%) is again higher than weight-only's alone (74.15%), reinforcing as on CIFAR-10. Activation sparsity does not reinforce here: joint additive's 74.13% is slightly

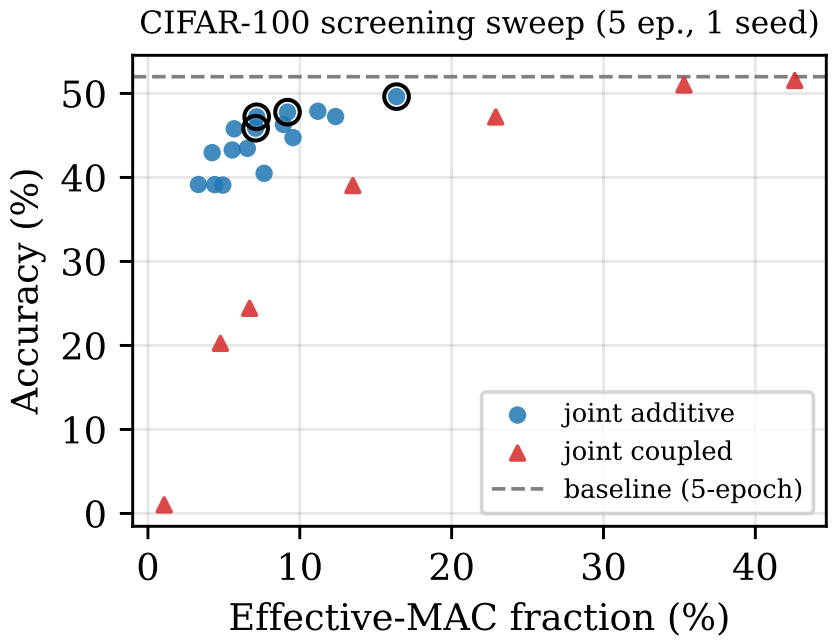


Figure 4: CIFAR-100 screening sweep (5 epochs, single seed). The same four candidates as CIFAR-10 (Figure 3) fall on the Pareto frontier here too (circled).

Table 9: Five-regime comparison on ResNet-18/CIFAR-100, 10 epochs, mean ± std over 3 seeds, coefficients selected by an independent CIFAR-100 sweep.

Regime	Accuracy	Weight Sparsity	Activation Sparsity	Effective-MAC Fraction
Baseline	$62.09 \pm 0.40\%$	$4.95 \pm 0.06\%$	$48.15 \pm 0.21\%$	$48.80 \pm 0.24\%$
Weight-only ($\lambda_w = 0.3$)	$59.41 \pm 0.68\%$	$82.43 \pm 0.12\%$	$49.25 \pm 0.40\%$	$8.87 \pm 0.10\%$
Activation-only ($\lambda_a = 0.05$)	$60.32 \pm 1.59\%$	$5.87 \pm 0.55\%$	$60.45 \pm 2.46\%$	$36.71 \pm 2.45\%$
Joint additive (weight-heavy)	$57.45 \pm 1.54\%$	$\mathbf{83.79 \pm 0.63\%}$	$66.40 \pm 3.41\%$	$5.97 \pm 0.60\%$
Joint coupled ($\lambda_c = 0.6$)	$42.14 \pm 2.94\%$	$70.54 \pm 0.56\%$	$\mathbf{79.10 \pm 1.82\%}$	$\mathbf{5.28 \pm 0.42\%}$

lower than activation-only's 77.04%, the opposite direction. The effective-MAC fraction still beats the naive independence estimate ($\approx 5.94\%$ vs. actual 4.00%), so some net reinforcement survives in effective-compute terms even with one marginal sparsity suppressed. Joint coupled reaches the *highest* accuracy of any regularized regime here ($98.97 \pm 0.07\%$, indistinguishable from baseline), the only setting in this paper where it is not dominated by joint additive, while still cutting effective-MAC more than $5\times$ ($44.10\% \rightarrow 8.11\%$, less than additive's 4.00% but at a smaller accuracy cost). Two confounds limit a clean comparison to CIFAR-10: LeNet's prunable set includes fully-connected layers, and MNIST's $> 98\%$ accuracy ceiling leaves much less room to observe degradation than CIFAR-10's 88% baseline.

D.2 ResNet-18/CIFAR-100

We keep the ResNet-18 architecture and increase only task difficulty (CIFAR-10's 10 classes to CIFAR-100's 100), repeating the same two-stage sweep independently rather than reusing CIFAR-10's coefficients. The Pareto-optimal region (Figure 4) selects the same four candidates as CIFAR-10's sweep, letting us compare at matched coefficients. Table 9 reports the full comparison.

Weight-heavy retains 92.5% of baseline accuracy while cutting effective-MAC $8.2\times$, a somewhat smaller retention fraction than CIFAR-10's 97.0%. Reinforcement is, if anything, clearer here: weight sparsity $+1.37$ points, activation sparsity $+5.95$ points, both larger and less noise-dominated

Table 10: Four-regime comparison on ResNet-18/SVHN, 10 epochs, mean ± std over 3 seeds, reusing the CIFAR-10 sweep-selected coefficients unchanged.

Regime	Accuracy	Weight Sparsity	Activation Sparsity	Effective-MAC Fraction
Baseline	$96.42 \pm 0.07\%$	$4.26 \pm 0.15\%$	$49.24 \pm 0.55\%$	$48.15 \pm 0.67\%$
Weight-only ($\lambda_w = 0.3$)	$96.38 \pm 0.04\%$	$92.08 \pm 0.16\%$	$52.31 \pm 1.00\%$	$3.75 \pm 0.01\%$
Activation-only ($\lambda_a = 0.05$)	$96.53 \pm 0.03\%$	$6.37 \pm 0.16\%$	$73.52 \pm 0.06\%$	$24.79 \pm 0.05\%$
Joint additive (weight-heavy)	$96.15 \pm 0.16\%$	$\mathbf{92.53 \pm 0.05}\%$	$80.22 \pm 0.29\%$	$\mathbf{1.68 \pm 0.06}\%$

Table 11: Joint coupled ($\lambda_c = 0.8$) on SVHN, 10 epochs, ten seeds, each generated independently through the standard pipeline – reported this way because the outcomes are bimodal rather than normally distributed around a mean.

Seed	0	1	2	3	4	5	6	7	8	9
Outcome	collapsed	collapsed	converged	collapsed	collapsed	collapsed	converged	collapsed	collapsed	collapsed
Accuracy	19.59%	19.59%	94.61%	19.59%	19.59%	19.59%	94.63%	19.59%	19.59%	19.59%

than the CIFAR-10 comparisons. The naive effective-MAC estimate (6.95%) exceeds the actual joint additive result (5.97%) by 14.1% relatively, larger than CIFAR-10's 9.6%. Recalibrated to $\lambda_c = 0.6$ (matching weight-heavy's effective-MAC), joint coupled costs 15.31 points more than joint additive, a larger gap than CIFAR-10's 14.12; it is also more fragile at high λ_c ($\lambda_c = 1.0$ collapses to 1.02% accuracy, chance level, versus CIFAR-10's degraded-but-not-collapsed 53.47% at the same coefficient) – together with the MNIST result in the opposite direction, indicating the coupled objective's relative disadvantage scales with task difficulty.

D.3 ResNet-18/SVHN

We hold architecture and class count fixed at 10, switching only the image domain, and reuse the CIFAR-10 sweep-selected coefficients unchanged. Table 10 reports baseline, weight-only, activation-only, and joint additive at the full protocol; joint coupled is discussed separately below since a single mean ± std would misrepresent its bimodal outcomes.

All four regimes cost less than 0.3 accuracy points relative to baseline; joint additive's $92.53\%/80.22\%$ weight/activation sparsity costs only 0.27 points, the smallest accuracy cost for the highest sparsity observed on any dataset in this paper. Reinforcement is clear on both axes, especially activation sparsity (80.22% vs. activation-only's 73.52%, a 6.70-point gap). The effective-MAC fraction (1.68%) is the lowest achieved anywhere in this paper.

Joint coupled at the reused $\lambda_c = 0.8$ is bimodal, not merely noisy. Across ten seeds run independently through the standard pipeline, two converge normally and eight collapse to an identical low accuracy (Table 11).[1]

Two of ten seeds converge to 94.6% accuracy (mean weight/activation sparsity $76.5\%/82.7\%$, effective-MAC 3.36%). The other eight land at 19.59–19.64% – not arbitrary: SVHN's test set is class-imbalanced, with digit "1" the single most frequent label at 19.59% of test examples. The collapsed seeds have settled into always predicting the majority class (99.4% activation sparsity, 0.21% effective-MAC, numbers that would look like an extraordinary success without the accompanying accuracy). We test four hypotheses about this instability directly. First, sweeping λ_c downward ($\{0.2, \ldots, 0.7\}$, four seeds each) does not eliminate it (Table 12): collapse persists at every coefficient, no monotonic trend.

Second, the two outcomes are not distinguishable from initialization: initial weight norm, per-element-normalized Hoyer values, and first-step gradient norms (measured via a separate throwaway forward-backward pass so as not to perturb the measured trajectory) show full overlap between eventually-converged and eventually-collapsed seeds. Third, early training is equally uninformative: at epochs 1 and 3, both groups sit at the same ≈ 18.5–19.1% training accuracy. Fourth, and most surprising, the outcome is not a fully reproducible function of the nominal seed: re-running seed 4

[1] Supersedes an earlier six-seed report whose seed-0 entry (90.32%, "converged") we found was a hardcoded value carried over from an earlier, undocumented run rather than one generated by the same pipeline as its other five seeds; we discard it and replace the whole table with a fresh, uniformly-generated ten-seed run.

Table 12: Joint coupled accuracy across a λ_c downward sweep on SVHN, 10 epochs, four seeds per coefficient. Bimodal collapse persists at every coefficient tested, down to $\lambda_c = 0.2$.

λ_c	Seed 0	Seed 1	Seed 2	Seed 3	Collapsed
0.2	95.87%	19.59%	19.62%	19.59%	3/4
0.3	93.39%	19.59%	95.53%	19.59%	2/4
0.4	19.64%	95.33%	94.98%	19.59%	2/4
0.5	95.35%	19.59%	95.23%	93.73%	1/4
0.6	19.59%	19.59%	19.59%	83.41%	3/4
0.7	19.59%	19.59%	19.60%	19.59%	4/4

Table 13: Joint coupled ($\lambda_c = 0.8$) on SVHN, 10 epochs, three reruns of the two seeds previously documented as outcome-unstable, with bit-deterministic execution forced on. Both are now perfectly reproducible.

Seed	Rep 1	Rep 2	Rep 3
2 (previously: 1 converged / 2 collapsed across 3 reruns)	19.5874%	19.5874%	19.5874%
4 (previously: 2 converged / 2 collapsed across 4 reruns)	90.5155%	90.5155%	90.5155%

four times gives two converged and two collapsed outcomes, and seed 2 similarly splits across three re-runs. `torch.manual_seed` fixes the pseudo-random sequence, but PyTorch's default CUDA convolution backward kernels are not bit-deterministic, and even functionally-inert extra computation (an early, since-corrected version of our own instrumentation) was enough to flip which basin a run fell into. This is coefficient-independent, state-independent, and only partially seed-dependent – closer to execution-sensitive basin selection than ordinary seed-dependent bistability. We still think SVHN's class imbalance plausibly shapes which basin exists to fall into, but the coefficient- and state-independence rules out a fixable calibration artifact as the full explanation.

Forcing deterministic execution confirms the nondeterminism hypothesis directly. We repeat seeds 2 and 4 – the two seeds documented above as flipping between outcomes across independent reruns – three times each with `torch.use_deterministic_algorithms(True)`, `cudnn.deterministic=True`, `cudnn.benchmark=False`, and `CUBLAS_WORKSPACE_CONFIG` set (Table 13). Both seeds are now perfectly stable: seed 2 collapses identically all three times (19.5874%, matching to four decimal places) and seed 4 converges identically all three times (90.5155%). This is exactly what the nondeterminism hypothesis predicts and directly rules out any remaining possibility that the seed-level irreproducibility reflects a bug in our own pipeline rather than genuine execution-level nondeterminism in the underlying CUDA kernels.

Learning-rate warmup and an alternative optimizer both fail to prevent collapse. We test two further interventions, each targeting a different candidate mechanism a reviewer might expect to help: a 2-epoch linear warmup (in case the instability is an early-training artifact of a too-large initial learning rate) and SGD with momentum in place of Adam (in case Adam's per-parameter adaptive normalization specifically contributes to the instability), both at $\lambda_c = 0.8$, ten seeds each. Neither helps: warmup leaves $10/10$ seeds collapsed, and SGD leaves $10/10$ collapsed – both at or above the untreated baseline's $8/10$, with SGD notably not converging on a single seed of the ten tested. Table 14 summarizes both against the baseline and the gradient-clipping result from Appendix H.

If anything, both new interventions perform slightly *worse* than doing nothing, though at $n = 10$ this is not a large enough sample to treat the exact rate as meaningful beyond "does not help." Combined with gradient clipping's null result, every training-recipe-level intervention we have tested now fails to move this instability, consistent with the confirmed execution-level nondeterminism above rather than a fixable optimization choice.

Table 14: Collapse rate on SVHN joint coupled ($\lambda_c = 0.8$, ten seeds each) across every intervention tested. None reduces the collapse rate below the untreated baseline.

Intervention	**Collapsed**
None (baseline)	$8/10$
Gradient clipping ($\|\nabla\| \leq 1$)	$8/10$
Learning-rate warmup (2 epochs)	$10/10$
SGD with momentum (in place of Adam)	$10/10$

Table 15: Five-regime comparison on MobileNetV2/CIFAR-10, 10 epochs, mean $\pm$ std over 3 seeds, reusing the ResNet-18/CIFAR-10 sweep-selected coefficients unchanged.

Regime	**Accuracy**	**Weight Sparsity**	**Activation Sparsity**	**Effective-MAC Fraction**
Baseline	$84.36 \pm 0.19\%$	$2.02 \pm 0.02\%$	$49.75 \pm 0.18\%$	$49.31 \pm 0.10\%$
Weight-only ($\lambda_w = 0.3$)	$80.37 \pm 0.50\%$	$71.38 \pm 0.21\%$	$49.40 \pm 0.10\%$	$14.54 \pm 0.07\%$
Activation-only ($\lambda_a = 0.05$)	$83.86 \pm 0.53\%$	$2.13 \pm 0.02\%$	$65.98 \pm 0.71\%$	$33.62 \pm 0.67\%$
Joint additive (weight-heavy)	$80.32 \pm 0.35\%$	$\mathbf{72.68 \pm 0.22\%}$	$66.38 \pm 0.18\%$	$10.30 \pm 0.11\%$
Joint coupled ($\lambda_c = 0.8$, unrecalibrated)	$46.03 \pm 3.86\%$	$13.69 \pm 0.41\%$	$\mathbf{97.62 \pm 0.12\%}$	$\mathbf{1.94 \pm 0.12\%}$

D.4 MOBILENETV2

We keep CIFAR-10 and the sweep-selected coefficients unchanged and switch only the network to a CIFAR-adapted MobileNetV2 (52 `Conv2d` weight tensors, 35 hooked activations, vs. ResNet-18's 20 and 17). Table 15 reports the results.

Joint additive retains 95.21% of baseline accuracy. Both marginal sparsities reinforce only slightly (weight $+1.30$, activation $+0.39$ points), and unlike every other dataset, the aggregate effective-MAC fraction does not beat the naive estimate (10.30% actual vs. 9.74% naive). At the reused $\lambda_c = 0.8$, joint coupled collapses to $46.03 \pm 3.86\%$ accuracy, a 34.29-point gap from joint additive, more than double ResNet-18/CIFAR-10's gap at the same coefficient; unlike SVHN this is not bimodal (all three seeds land in a continuous 42–51% range). A plausible explanation is architectural: the coupled loss sums over every paired weight-activation term, and MobileNetV2 has $2.6\times$ as many `Conv2d` weight tensors as ResNet-18, so the identical scalar λ_c plausibly exerts far more aggregate pressure.

We repeat the check on CIFAR-100, reusing CIFAR-100's own independently-swept coefficients ($\lambda_c = 0.6$) so only architecture varies within each dataset's row of a 2×2 grid. Table 16 reports the results.

Reinforcement now decisively clears the naive baseline (weight $+2.58$, activation $+3.20$ points; effective-MAC 10.02% actual vs. $\approx 10.84\%$ naive, opposite sign from CIFAR-10), mirroring the ResNet-18 pattern where task difficulty strengthens additive reinforcement largely independently of architecture. Joint coupled collapses to $10.96 \pm 6.64\%$ accuracy, barely above chance, with one seed at 1.63% against two partially converging to 14.7% and 16.55% – qualitatively different from MobileNetV2/CIFAR-10's continuous underperformance and closer to SVHN's bimodal instability, despite a completely different dataset/architecture combination. Reading the full 2×2 grid together (Tables 1, 9, 15, 16; summarized in Table 17) resolves what looks like a contradiction into two true statements about two different scales. **"Additive" here means**: the two architectures' *absolute* additive-coupled accuracy-point gaps increase by nearly identical amounts when task difficulty alone increases (ResNet-18: $14.12 \to 15.31$, $+1.19$; MobileNetV2: $34.29 \to 35.26$, $+0.97$) – task difficulty's absolute cost to the gap is architecture-independent, the signature of two factors whose effects simply sum rather than compound. **"Interacts destructively" here means**: the same $\sim$1-point absolute increment lands on architectures with wildly different baseline coupled-regime health, so *relative* accuracy retention (coupled accuracy over *baseline* accuracy, this paper's standard convention) drops by very different amounts – 13.3 points on ResNet-18 (CIFAR-10$\to$CIFAR-100) versus 33.7 points on MobileNetV2 – because MobileNetV2's coupled regime already retains proportionally less of baseline even at the easier task, so the identical absolute cost represents a proportionally much larger hit. The two factors are additive on the scale the coupled objective's own optimization difficulty operates on (absolute accuracy points lost to slow convergence, Section 4.2) and interactive on the

Table 16: Five-regime comparison on MobileNetV2/CIFAR-100, 10 epochs, mean ± std over 3 seeds, reusing CIFAR-100's own ResNet-18-swept coefficients ($\lambda_w = 0.3$, $\lambda_a = 0.05$, $\lambda_c = 0.6$) unchanged.

Regime	Accuracy	Weight Sparsity	Activation Sparsity	Effective-MAC Fraction
Baseline	$52.44 \pm 0.52\%$	$2.10 \pm 0.03\%$	$49.41 \pm 0.17\%$	$49.62 \pm 0.13\%$
Weight-only ($\lambda_w = 0.3$)	$47.60 \pm 0.50\%$	$70.41 \pm 0.20\%$	$49.06 \pm 0.09\%$	$15.10 \pm 0.08\%$
Activation-only ($\lambda_a = 0.05$)	$51.87 \pm 1.59\%$	$2.21 \pm 0.02\%$	$63.38 \pm 1.38\%$	$36.27 \pm 1.35\%$
Joint additive (weight-heavy)	$46.22 \pm 0.71\%$	$\mathbf{72.99 \pm 0.25\%}$	$\mathbf{66.58 \pm 0.40\%}$	$\mathbf{10.02 \pm 0.14\%}$
Joint coupled ($\lambda_c = 0.6$)	$10.96 \pm 6.64\%$	$11.74 \pm 2.33\%$	$97.49 \pm 0.94\%$	$1.99 \pm 0.63\%$

Table 17: The architecture × task-difficulty 2 × 2 grid, ResNet-18 vs. MobileNetV2 by CIFAR-10 vs. CIFAR-100 (all at each architecture's own sweep- or task-selected coefficients; Tables 1, 9, 15, 16). "Abs. Gap" is joint additive accuracy minus joint coupled accuracy, in points – additive across architectures. "Rel. Retention" is joint coupled accuracy / *baseline* accuracy (this paper's standard convention, Section 4.3) – interactive across architectures.

	CIFAR-10	CIFAR-100
ResNet-18 – Abs. Gap	14.12 pts	15.31 pts
ResNet-18 – Rel. Retention	81.2%	67.9%
MobileNetV2 – Abs. Gap	34.29 pts	35.26 pts
MobileNetV2 – Rel. Retention	54.6%	20.9%

scale a practitioner deciding whether to deploy the coupled objective would actually care about (how much of the achievable accuracy survives).

D.5 Recalibrating λ_c for MobileNetV2/CIFAR-100

The $\lambda_c = 0.6$ used above is CIFAR-100's own *task*-recalibrated coefficient (Appendix D.2), reused unchanged across the architecture change to MobileNetV2 – recalibrated for the harder task, but not for this architecture's $2.6\times$ larger number of paired weight-activation terms. We test whether an architecture-specific recalibration closes the gap: a screening sweep at 5 epochs, single seed (Table 18) over $\lambda_c \in \{0.05, 0.1, 0.15, 0.2, 0.23, 0.3, 0.4, 0.6\}$, matching the search to joint additive's own effective-MAC target (10.02%), selects $\lambda_c = 0.23$ (11.01% at 5 epochs, closest to target alongside 0.2; we take the marginally closer 0.23).

Table 19 reports the full protocol (10 epochs, three seeds) at $\lambda_c = 0.23$, alongside the reused-coefficient row from Table 16 for reference.

Recalibration substantially improves both accuracy and stability: accuracy rises from $10.96 \pm 6.64\%$ to $33.06 \pm 1.49\%$ (a 22.10-point improvement), and seed variance shrinks more than fourfold ($\pm 6.64 \rightarrow \pm 1.49$), moving from a high-variance, bimodal-flavored failure to ordinary, well-behaved seed-to-seed variation. This confirms the calibration-asymmetry mechanism concretely: MobileNetV2/CIFAR-100's most severe failure in this paper was substantially, though not entirely, an architecture-driven miscalibration rather than a fundamentally different or uncontrollable instability – unlike SVHN's, which persists across recalibration (Appendix D.3) and is not fixed by any intervention we have tested. A 33.06-vs-66.49% gap to joint additive remains even after recalibration (weight sparsity, in particular, stays far below joint additive's 72.99%), so this is a partial, not complete, resolution: the coupled objective's disadvantage at this architecture/task combination is calibration-driven in large part, but not solely.

D.6 Structured (Channel-Level) Sparsity

We replace the per-element weight-Hoyer measure with a structured analogue applying the identical formula to per-output-channel L_2 norms, measuring weight sparsity as the fraction of near-zero-norm channels. Everything else (architecture, dataset, coefficients, protocol) is identical to Table 1. Table 6 (main text) reports the results.

Table 18: λ_c screening for MobileNetV2/CIFAR-100, 5 epochs, single seed. $\lambda_c = 0.6$ is the reused, unrecalibrated coefficient from Table 16; $\lambda_c = 0.23$ (bold) most closely matches joint additive's 10.02% effective-MAC target and is carried forward.

λ_c	Accuracy	Weight Sp.	Act. Sp.	Eff.-MAC
0.05	37.75%	4.13%	60.85%	37.02%
0.1	31.65%	5.84%	75.45%	22.88%
0.15	28.64%	6.85%	82.45%	16.09%
0.2	23.39%	7.99%	87.81%	11.01%
0.23	21.90%	8.71%	87.99%	**10.81%**
0.3	19.75%	8.56%	91.75%	7.31%
0.4	15.62%	9.07%	94.39%	4.85%
0.6 (reused, unrecalibrated)	3.87%	7.98%	96.98%	2.82%

Table 19: MobileNetV2/CIFAR-100 joint coupled, recalibrated ($\lambda_c = 0.23$) vs. reused ($\lambda_c = 0.6$, from Table 16), 10 epochs, mean $\pm$ std over 3 seeds.

Configuration	Accuracy	Weight Sparsity	Activation Sparsity	Effective-MAC Fraction
Joint additive (for reference)	$46.22 \pm 0.71\%$	$72.99 \pm 0.25\%$	$66.58 \pm 0.40\%$	$10.02 \pm 0.14\%$
Joint coupled, $\lambda_c = 0.6$ (reused)	$10.96 \pm 6.64\%$	$11.74 \pm 2.33\%$	$97.49 \pm 0.94\%$	$1.99 \pm 0.63\%$
Joint coupled, $\lambda_c = 0.23$ (recalibrated)	$\mathbf{33.06 \pm 1.49}\%$	$11.60 \pm 0.37\%$	$90.07 \pm 1.17\%$	$8.72 \pm 1.04\%$

Joint additive's channel sparsity (60.32%) exceeds weight-only's (48.82%) by 11.50 points, more than $17\times$ CIFAR-10's unstructured gap, and clears significance decisively ($p = 0.0006$); activation sparsity reinforces by 13.25 points ($p = 0.0067$). The naive effective-MAC estimate ($\approx 18.89\%$) exceeds the actual result (10.81%) by 42.8% relatively, more than four times CIFAR-10's unstructured 9.6%. Joint coupled collapses to $49.56 \pm 10.94\%$ accuracy while reaching almost no channel sparsity (1.62% vs. additive's 60.32%), pushing sparsity almost entirely into activations (93.99%) – a different failure signature from the unstructured case, where coupled still reached substantial (71.86%) weight sparsity. (Section 4.3 connects the size of this reinforcement gap to Proposition 2: a fully-zeroed channel deterministically zeroes its entire downstream activation map, rather than merely shrinking its variance the way a single zeroed weight element does.)

We repeat the check on CIFAR-100 (Table 20), reusing CIFAR-100's own $\lambda_c = 0.6$.

Reinforcement is present (channel $+8.16$, activation $+13.34$ points) but weaker and less statistically clean than CIFAR-10 (channel-sparsity gap $p = 0.082$, not significant) – the *opposite* of the unstructured pattern, where reinforcement is consistently clearer on CIFAR-100. Joint coupled fails more severely on the harder task ($21.74 \pm 8.57\%$ accuracy, channel sparsity 0.39%).

Layer-wise analysis. We instrument every one of ResNet-18's 20 `Conv2d` layers individually (single seed, 10-epoch protocol) rather than only the network-wide mean. Figure 5 shows the full breakdown. The per-layer channel- and activation-sparsity reinforcement gaps (joint additive minus the respective solo regime) are positively correlated across layers ($r = 0.51$): layers pruned disproportionately more also gain disproportionately more activation sparsity. The channel-sparsity gap is concentrated in the early/middle two-thirds of the network (mean $+14.97$, $+17.32$ points) and collapses to near zero in the final third ($+0.81$); activation-sparsity reinforcement is comparatively even across depth.

The coupled formulation's near-total channel-sparsity failure is concentrated, not diffuse: of the 20 layers, 18 reach exactly 0% channel sparsity, and only `layer2.0.conv1` (7.8%, the first channel-doubling, stride-2 convolution) and `layer4.1.conv2` (20.7%, the final convolution before the classifier) achieve any structured sparsity at all, while activation sparsity stays uniformly high (86.7–99.9%) everywhere. We did not investigate why these two layers differ from the other eighteen beyond noting their structural distinctiveness (a channel-count transition and direct classifier adjacency).

The channel-level granularity tested here is the coarsest end of a spectrum that also includes hardware-supported semi-structured patterns such as $2{:}4$ sparsity (2 nonzero of every 4 contiguous weights) on NVIDIA Ampere and Hopper GPUs. Proposition 2's deterministic-zeroing argument – a fully-zeroed group's pre-activation contribution is exactly zero rather than merely lower-variance – is

Table 20: Five-regime comparison with structured (channel-level) weight sparsity on ResNet-18/CIFAR-100, 10 epochs, mean ± std over 3 seeds, reusing CIFAR-100's own independently-swept coefficients ($\lambda_c = 0.6$) unchanged.

Regime	Accuracy	Channel Sparsity	Activation Sparsity	Effective-MAC Fraction
Baseline	$62.08 \pm 0.14\%$	$0.00 \pm 0.00\%$	$48.17 \pm 0.20\%$	$51.23 \pm 0.25\%$
Weight-only ($\lambda_w = 0.3$)	$60.69 \pm 1.59\%$	$44.52 \pm 3.80\%$	$56.49 \pm 2.48\%$	$25.68 \pm 3.08\%$
Activation-only ($\lambda_a = 0.05$)	$60.74 \pm 1.30\%$	$0.00 \pm 0.00\%$	$60.37 \pm 2.34\%$	$38.97 \pm 2.35\%$
Joint additive (weight-heavy)	$58.24 \pm 1.68\%$	$\mathbf{52.69 \pm 3.14\%}$	$\mathbf{73.71 \pm 1.28\%}$	$\mathbf{15.16 \pm 1.46\%}$
Joint coupled ($\lambda_c = 0.6$, unrecalibrated)	$21.74 \pm 8.57\%$	$0.39 \pm 0.44\%$	$91.39 \pm 2.74\%$	$7.79 \pm 2.48\%$

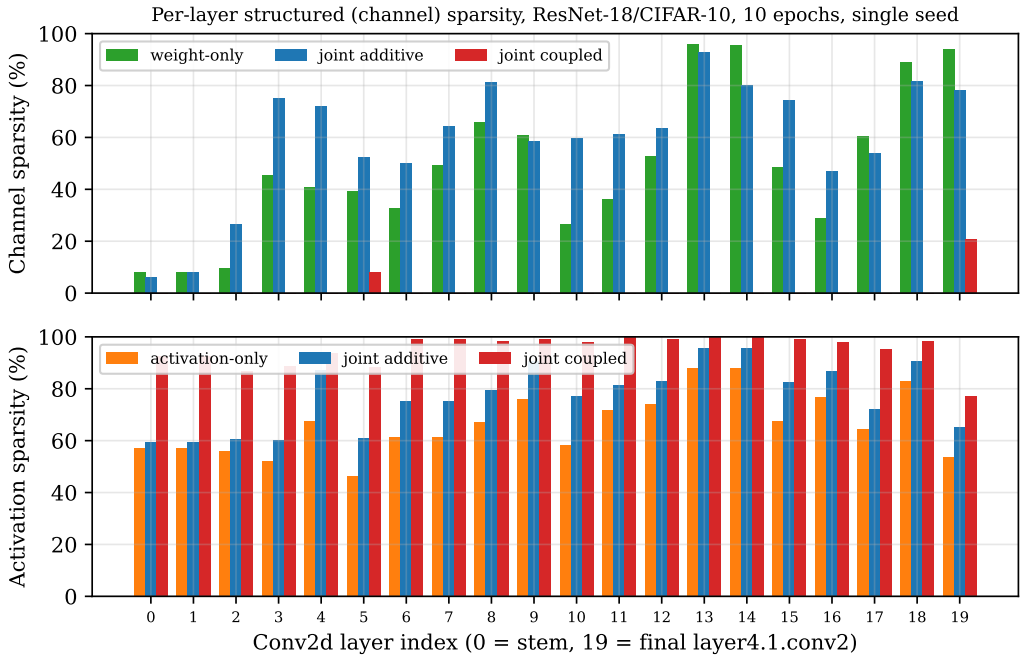


Figure 5: Per-layer channel sparsity (top) and activation sparsity (bottom) across all 20 `Conv2d` layers of ResNet-18/CIFAR-10, single seed, 10 epochs. Joint coupled (red) reaches near-zero channel sparsity everywhere except layers 5 and 19, while its activation sparsity (bottom, red) stays uniformly near-saturated at every layer.

granularity-agnostic in principle: it applies equally to a zeroed output channel, a zeroed 2:4 block, or a zeroed individual weight, predicting reinforcement should strengthen monotonically as granularity coarsens from unstructured through block patterns to full channels, matching the ordering we in fact observe between our two tested extremes (Table 1 vs. Table 6). We have not tested an intermediate block-sparse granularity directly, so this is a testable prediction of the mechanism rather than a reported result.

E Is Reinforcement Specific to Hoyer Regularization? An L1 Control

Section 4.1 shows weight and activation sparsity reinforce under joint Hoyer regularization. Is this specific to the Hoyer measure's scale-invariant functional form, or a more general property of jointly sparsifying both operands under one training loop? We test this with the simplest alternative: a per-element-normalized L1 penalty, $\lambda_w \sum_l \mathrm{mean}(|W_l|)$ and $\lambda_a \sum_l \mathrm{mean}(|A_l|)$ (same per-element normalization convention as Hoyer, so one coefficient exerts comparable pressure across layer sizes, but without Hoyer's L_1^2/L_2 scale-invariant ratio), on ResNet-18/CIFAR-10, all other settings identical to Table 1.

Table 21: L1 coefficient screening, ResNet-18/CIFAR-10, 5 epochs, single seed. Weight-side L1 barely responds until far larger coefficients than Hoyer requires; activation-side L1 responds smoothly. Bold rows are carried forward.

λ_w	Accuracy	Weight Sp.	λ_a	Accuracy	Act. Sp.
0.005	83.24%	4.34%	0.005	83.46%	48.25%
0.01	83.81%	4.42%	0.01	82.76%	49.55%
0.02	83.45%	4.52%	0.02	81.91%	54.03%
0.05	83.50%	4.63%	**0.05**	78.73%	**62.74%**
0.1	82.85%	5.33%	0.1	76.34%	69.77%
0.3	83.37%	7.33%			
0.5	83.33%	9.63%			
1.0	83.22%	16.78%			
2.0	82.85%	30.09%			
5.0	80.83%	**59.66%**			

Table 22: L1 control on ResNet-18/CIFAR-10, 10 epochs, mean ± std over 3 seeds ($\lambda_w = 5.0$, $\lambda_a = 0.05$), vs. the Hoyer rows from Table 1.

Regime	Accuracy	Weight Sparsity	Activation Sparsity	Effective-MAC Fraction
Weight-only, Hoyer ($\lambda_w = 0.3$)	$87.34 \pm 0.13\%$	$86.08 \pm 0.43\%$	$48.97 \pm 0.19\%$	$7.08 \pm 0.23\%$
Weight-only, L1 ($\lambda_w = 5.0$)	$\mathbf{88.24 \pm 0.28}\%$	$74.01 \pm 0.16\%$	$54.20 \pm 0.48\%$	$11.90 \pm 0.03\%$
Activation-only, Hoyer ($\lambda_a = 0.05$)	$87.99 \pm 0.21\%$	$5.70 \pm 0.36\%$	$63.08 \pm 1.75\%$	$34.57 \pm 1.75\%$
Activation-only, L1 ($\lambda_a = 0.05$)	$87.92 \pm 0.83\%$	$5.27 \pm 0.36\%$	$\mathbf{67.67 \pm 0.02}\%$	$30.53 \pm 0.44\%$
Joint additive, Hoyer	$86.33 \pm 0.35\%$	$86.75 \pm 0.10\%$	$67.72 \pm 2.15\%$	$4.63 \pm 0.12\%$
Joint additive, L1	$86.29 \pm 0.96\%$	$81.80 \pm 0.73\%$	$79.96 \pm 0.46\%$	$\mathbf{4.03 \pm 0.23}\%$

L1 requires far larger coefficients than Hoyer to induce comparable sparsity, and behaves differently on the two operands. A screening sweep at 5 epochs, single seed (Table 21) finds activation-side L1 responds smoothly to its coefficient (a familiar shape), but weight-side L1 barely moves sparsity across two orders of magnitude ($\lambda_w \in [0.005, 0.1]$: weight sparsity $4.3\% \to 5.3\%$, indistinguishable from an untrained network's natural near-zero-weight rate) before finally responding at far larger coefficients ($\lambda_w = 5.0$: 59.66% at 5 epochs). We select $\lambda_w = 5.0$ (roughly $16\times$ Hoyer's $\lambda_w = 0.3$) and $\lambda_a = 0.05$ (identical to Hoyer's) for the full protocol.

Table 22 reports the full protocol (10 epochs, three seeds). At the full protocol, weight-only L1's sparsity rises well beyond its 5-epoch screening value (74.01% vs. 59.66% at 5 epochs), reaching a more competitive, though still lower, operating point than Hoyer weight-only (86.08%).

Reinforcement is not specific to Hoyer regularization – if anything, it is quantitatively stronger with L1 at these coefficients. Joint L1's weight sparsity (81.80%) exceeds weight-only L1's (74.01%) by $+7.79$ points, more than an order of magnitude larger than Hoyer's own $+0.67$-point gap at $n = 5$ (Table 1). Activation sparsity reinforces by $+12.29$ points (79.96% vs. 67.67%), also larger than Hoyer's $+4.64$ points. The naive independence estimate for joint L1's effective-MAC fraction is $(100 - 74.01\%) \times (100 - 67.67\%) \approx 8.40\%$; the actual result ($4.03\%$) beats this by 52.0% relatively, more than five times Hoyer's 9.9% reduction at $n = 5$. This is evidence the reinforcement phenomenon is a general property of jointly sparsifying weight and activation tensors under a shared training loop, not an artifact specific to Hoyer's scale-invariant functional form. It is not, however, a strictly better alternative: L1 reaches a different, overall less favorable operating point (lower peak weight sparsity, and requiring a coefficient roughly $16\times$ larger than Hoyer's to get there at all), and weight-side L1's near-total insensitivity to its coefficient across two orders of magnitude before finally responding at extreme values is itself a notable qualitative difference from Hoyer's smooth, well-behaved response across a comparable range – consistent with Hoyer's smoothly-vanishing-near-zero

Table 23: Weight-only pushed to match joint additive's effective-MAC fraction, ResNet-18/CIFAR-10, 10 epochs, mean ± std over 5 seeds. Joint additive row reproduced from Table 1. p is Welch's t-test, uncorrected (single comparison, not part of the paper's main significance family).

Regime	Accuracy	Weight Sp.	Activation Sp.	Effective-MAC
Weight-only, matched ($\lambda_w = 0.7$)	$\mathbf{86.63 \pm 0.26}\%$	$91.10 \pm 0.28\%$	$49.37 \pm 0.47\%$	$4.48 \pm 0.11\%$
Joint additive ($\lambda_w = 0.3, \lambda_a = 0.05$)	$86.33 \pm 0.35\%$	$86.75 \pm 0.10\%$	$67.72 \pm 2.15\%$	$4.63 \pm 0.12\%$

gradient interacting more favorably with Adam's per-parameter adaptive normalization than L1's constant-magnitude subgradient.

F Does Joint Training Beat a Matched-Compute Solo Penalty?

Every reinforcement comparison so far holds the *coefficient* fixed and asks whether joint training pushes sparsity further than a solo penalty at that same coefficient. This leaves open a different, more practical question: if a practitioner only cares about the accuracy reachable at a given effective-MAC fraction, does joint additive's reinforcement translate into an advantage over simply turning up a single solo penalty until it reaches that same effective-MAC fraction alone? The two questions are not the same – joint additive could legitimately reinforce both marginal sparsities at matched coefficients while still landing on a worse accuracy-compute frontier than a solo penalty is capable of reaching on its own.

We screen $\lambda_w \in \{0.4, 0.5, 0.6, 0.7, 0.8, 1.0\}$ for weight-only alone (5 epochs, single seed) to find the coefficient whose effective-MAC fraction most closely matches joint additive's own 4.63% (Table 1); $\lambda_w = 0.7$ reaches 4.62% at 10 epochs, essentially exact. We then validate this configuration at the full protocol ($n = 5$ seeds, matching Table 1's upgraded sample size for this specific comparison).

The two effective-MAC fractions are statistically indistinguishable ($t = -1.84$, $p = 0.103$), confirming the match; the two accuracies are also statistically indistinguishable ($t = 1.37$, $p = 0.212$), but the point estimate favors the *solo* penalty by $+0.30$ points, not joint additive. At matched compute, joint additive's reinforcement – real at the level of marginal sparsities and covariance (Section 4.1) – does not demonstrate an accuracy advantage over simply turning up a single weight-only penalty until it reaches the same effective-MAC fraction on its own; if anything, the single-seed screen and the $n = 5$ validation both point the other way. The two regimes reach their shared compute target through very different sparsity patterns (weight-only-matched: 91.10% weight / 49.37% activation, close to baseline's own activation sparsity, i.e. almost all of its compute reduction comes from the weight side; joint additive: 86.75% weight / 67.72% activation, a more balanced split) – reinforcement changes *how* the compute reduction is achieved without demonstrating that it improves *how much* accuracy that reduction costs. We report this as a direct answer to whether reinforcement is practically load-bearing for a practitioner optimizing accuracy at a fixed compute target, rather than assuming a phenomenon documented at matched coefficients must also confer an advantage at matched compute: on this single architecture and dataset, it does not, at $n = 5$ seeds power to detect a ± 0.3-point effect.

G Directly Measuring the Gradient-Stalling Mechanism

Section 4.2 infers the coupled objective's slow convergence from training-budget and intervention outcomes (log-space reformulation, gradient clipping; Appendix H) rather than from a direct measurement of the hypothesized mechanism: for the product term $\lambda_c \sum_l H(W_l)H(A_l)$, $\partial/\partial W_l$ of layer l's term equals $H(A_l) \cdot \partial H(W_l)/\partial W_l$ by the product rule, so the gradient reaching W_l is scaled by the activation factor's *current* value, and vice versa – as either factor shrinks toward its regularizer's target, it throttles the other's remaining gradient signal, independent of how far that other factor still is from its own target. We instrument one standard training run (ResNet-18/CIFAR-10, joint coupled, $\lambda_c = 0.8$, seed 0) to log this directly: every 20 steps, for three tracked weight-activation pairs, we compute $H(W_l)$, $H(A_l)$, the isolated gradient norm of the full coupled term with respect to W_l (via `autograd.grad` on the term alone, before the real backward pass), and the same gradient with $H(A_l)$ held out (i.e. $\|\partial H(W_l)/\partial W_l\|$ alone).

Table 24: Direct gradient-trajectory measurement, ResNet-18/CIFAR-10, joint coupled ($\lambda_c = 0.8$), 10 epochs, single seed (72.08% final accuracy, consistent with Table 1's $72.17 \pm 1.59\%$). Ratios are step-0-to-step-3900 (full training). "Own decline" is $H(W_l)$'s own regularizer-driven progress in isolation ($\|\partial H(W_l)/\partial W_l\|$ shrinking as W_l approaches its target); "partner throttle" is $H(A_l)$'s decline, the factor multiplying that gradient by the product rule.

Layer	$H(W)$ decline	$H(A)$ decline	Own decline	Partner throttle	Coupled-gradient decline
Early (pair 1)	$2.89\times$	$4.72\times$	$2.11\times$	$4.72\times$	$9.93\times$
Mid (pair 10)	$8.59\times$	$22.93\times$	$3.18\times$	$22.93\times$	$72.95\times$
Late (pair 16)	$7.21\times$	$12.97\times$	$3.86\times$	$12.97\times$	$50.07\times$

An honest complication in pair selection. The approximate pairing scheme (Section 4.2) matches each of the network's 20 weight tensors to the nearest of 17 hooked activations by relative depth; because weights outnumber hooked activations, several trailing weight tensors are paired with an activation that is their *own* direct output rather than an independent downstream one, for which $H(A_l)$ genuinely depends on W_l and the product-rule factorization above does not isolate cleanly. We check all 20 pairs directly (computing $\partial H(A_l)/\partial W_l$ via `autograd.grad` with `allow_unused=True`) and find 6 of 20 (pairs 0, 12, 13, 17, 18, 19) are such self-dependent pairings; we select three verified-independent pairs spanning the network's depth for tracking – pair 1 (early), pair 10 (mid), pair 16 (late) – rather than the naive first/middle/last-index choice, which would have silently picked a self-dependent pair.

Table 24 confirms the mechanism exactly rather than approximately: at every logged step for all three pairs, the isolated coupled-term gradient on W_l equals $H(A_l)$ times $\|\partial H(W_l)/\partial W_l\|$ to eight decimal places (an exact identity, since this is the product rule applied to measured quantities, not a statistical fit). The practically important result is which factor *dominates* the gradient's overall decline over training: in all three layers, the partner's throttling (4.7–$22.9\times$) exceeds the operand's own regularizer-driven progress (2.1–$3.9\times$) by a wide margin, and the effect is more pronounced in deeper layers, where activations sparsify much faster than weights (mid layer: $H(A)$ falls $22.93\times$ against $H(W)$'s $8.59\times$). This means that for most of training, the weight-side gradient from the coupled term is small primarily because the *activation* side has already become sparse, not because the weight itself is near its own target – precisely the stalling dynamic Section 4.2 infers indirectly from convergence speed, now confirmed as an exact, measured property of the actual training trajectory. Code for this instrumentation (40 lines on top of the standard training loop) is available alongside the training code.

H Testing Interventions Against the Coupled Objective's Failure Modes

Section 5 summarizes three interventions tested directly against the coupled objective's two distinct failure modes (slow, product-loss-limited convergence on CIFAR-10; bimodal collapse on SVHN) rather than leaving them as untested speculation. All three reuse the standard 10-epoch, three-seed protocol (ten seeds for SVHN, to estimate a collapse rate) and the sweep-selected $\lambda_c = 0.8$ configuration except where noted.

Log-space reformulation. We replace the coupled term with $\lambda_c \sum_l [\log(H(W_l) + \epsilon) + \log(H(A_l) + \epsilon)]$ ($\epsilon = 10^{-3}$), which shares the product's fixed points (both factors driven toward zero) but removes the product rule's multiplicative gradient coupling: $\partial/\partial W_l$ of the log-sum term scales with $1/H(W_l)$ alone, not with $H(A_l)$'s current value. Because this changes the loss's absolute scale completely, $\lambda_c = 0.8$ is not meaningful in log-space; we screen $\lambda_c \in \{0.001, 0.003, 0.01, 0.03, 0.1\}$ at 5 epochs, single seed (Table 25) and select $\lambda_c = 0.03$, whose effective-MAC fraction (4.72%) most closely matches the product-coupled target (4.64%).

Table 26 reports the full protocol at this coefficient, alongside the standard product-coupled and additive rows from Table 1 for reference.

Log-space reaches a substantially lower effective-MAC fraction than standard product-coupled (2.70% vs. 3.93%, a 31% relative reduction) at statistically indistinguishable accuracy ($70.61 \pm 3.22\%$ vs. $72.17 \pm 1.59\%$; the accuracy point estimate is nominally lower but well within the combined

Table 25: Log-space coupled objective, screening sweep on ResNet-18/CIFAR-10, 5 epochs, single seed. $\lambda_c = 0.03$ (bold) is carried forward to the full protocol.

λ_c	Accuracy	Weight Sp.	Act. Sp.	Eff.-MAC
0.001	83.11%	5.15%	47.86%	48.59%
0.003	83.25%	7.93%	49.90%	45.03%
0.01	78.81%	41.65%	69.67%	20.49%
0.03	60.34%	74.33%	91.06%	**4.72%**
0.1	24.44%	76.85%	95.64%	3.03%

Table 26: Log-space coupled objective ($\lambda_c = 0.03$) on ResNet-18/CIFAR-10, 10 epochs, mean $\pm$ std over 3 seeds, vs. the product-coupled and additive rows from Table 1.

Regime	Accuracy	Weight Sparsity	Activation Sparsity	Effective-MAC Fraction
Joint additive	$86.40 \pm 0.40\%$	$86.75 \pm 0.13\%$	$67.57 \pm 2.38\%$	$4.64 \pm 0.12\%$
Joint coupled, log-space ($\lambda_c = 0.03$)	$70.61 \pm 3.22\%$	$84.54 \pm 0.41\%$	$92.32 \pm 0.69\%$	$\mathbf{2.70 \pm 0.22}\%$
Joint coupled, product (standard, $\lambda_c = 0.8$)	$72.17 \pm 1.59\%$	$71.86 \pm 1.03\%$	$84.30 \pm 1.22\%$	$3.93 \pm 0.27\%$

variance, which is itself larger than the product form's, ± 3.22 vs. ± 1.59). It reaches this via a different route than the product form: substantially higher activation sparsity (92.32% vs. 84.30%) and slightly higher weight sparsity (84.54% vs. 71.86%). Neither form closes the gap to additive's 86.40%.

Gradient clipping. We apply $\|\nabla\| \leq 1$ gradient-norm clipping to standard product-coupled training ($\lambda_c = 0.8$), no other change. Table 27 reports the result.

Clipping raises accuracy by 3.02 points (75.19% vs. 72.17%), closing roughly a fifth of the 14.12-point gap to additive, at a small effective-MAC cost (4.52% vs. 3.93%); weight and activation sparsity are each slightly lower than the unclipped case. The gain is directionally consistent across all three seeds (seed-level accuracies 75.61%, 77.21%, 72.76%, all at or above the unclipped mean) but, at $n = 3$, does not on its own establish significance to the same standard used elsewhere in this paper.

Gradient clipping against SVHN's bimodal instability. We repeat the identical $\|\nabla\| \leq 1$ clipping on SVHN's joint-coupled regime ($\lambda_c = 0.8$) across ten seeds, the same protocol used for the unclipped bimodality check (Appendix D.3, Table 11).

$8/10$ seeds collapse – exactly the same rate as unclipped (Table 11), and the two converged seeds reach accuracy (95.01%, 94.94%) matching the unclipped converged seeds (94.61%, 94.63%) closely. Gradient clipping has no detectable effect on either the collapse rate or the outcome quality conditional on convergence, direct evidence against gradient scale as the driver of SVHN's instability, consistent with the coefficient- and state-independence already established (Appendix D.3).

I Combining Additive and Coupled Terms (Hybrid Objective)

Does the product term contribute anything useful on top of the additive sum, even though it fails badly alone? We test a hybrid objective, $\mathcal{L}_{\text{CE}} + \lambda_w \sum_l H(W_l) + \lambda_a \sum_l H(A_l) + \lambda_c \sum_l H(W_l) \cdot H(A_l)$, using the additive coefficients unchanged and adding the coupled term at two coefficients found well-behaved in a short calibration check ($\lambda_c \in \{0.2, 0.4\}$; the full $\lambda_c = 0.8$ alone was too aggressive once stacked on the additive terms). Table 29 reports the full protocol.

Both hybrid configurations *dominate* the pure coupled regime on every metric simultaneously: higher accuracy, higher weight sparsity, and lower (better) effective-MAC (3.39%/2.64% vs. 3.93%). Relative to pure additive, the hybrid trades accuracy for additional compute reduction pure additive alone does not reach: $\lambda_c = 0.2$ costs 2.58 points for a further 27% relative cut in effective-MAC, and $\lambda_c = 0.4$ costs 5.78 points for a further 43% cut. The coupled term's failure alone is thus specifically a failure as a *standalone* objective, not evidence it carries no useful signal: on top of an already-well-optimized additive baseline, at a modest coefficient, it extracts real additional compute reduction at a moderate, controllable accuracy cost.

Table 27: Standard product-coupled objective with gradient clipping ($\|\nabla\| \leq 1$) on ResNet-18/CIFAR-10, 10 epochs, mean $\pm$ std over 3 seeds, vs. the unclipped row from Table 1.

Regime	**Accuracy**	**Weight Sp.**	**Act. Sp.**	**Eff.-MAC**
Joint coupled, unclipped	$72.17 \pm 1.59\%$	$71.86 \pm 1.03\%$	$84.30 \pm 1.22\%$	$3.93 \pm 0.27\%$
Joint coupled, clipped	$\mathbf{75.19 \pm 1.84}\%$	$71.49 \pm 1.53\%$	$81.00 \pm 1.93\%$	$4.52 \pm 0.31\%$

Table 28: Joint coupled with gradient clipping ($\|\nabla\| \leq 1$) on SVHN, 10 epochs, ten seeds. Collapse rate is identical to the unclipped baseline (Table 11).

Seed	0	1	2	3	4	5	6	7	8	9
Outcome	converged	collapsed	collapsed	collapsed	collapsed	collapsed	collapsed	collapsed	converged	collapsed
Accuracy	95.01%	19.59%	19.59%	19.59%	19.59%	19.59%	19.59%	19.59%	94.94%	19.59%

J Extended Training Budgets

For every dataset/architecture combination where we tested more than one training budget, the main text reports only the 10-epoch headline result, used for cross-comparison throughout, and summarizes the longer-training findings inline (Figure 2). This appendix reports those additional checkpoints in full.

J.1 ResNet-18/CIFAR-10 at 20 and 50 Epochs

Table 30 re-runs all five Table 1 regimes for 20 epochs instead of 10, same coefficients, three seeds each.

Every regime improves with more training, but the improvement is highly uneven. Joint additive's accuracy rises by 3.61 points, while joint coupled's rises by 11.74 points – more than three times as much. The additive-coupled accuracy gap shrinks from 14.12 points at 10 epochs to 6.10 points at 20. At 10 epochs coupled's effective-MAC fraction (3.93%) was lower (better) than additive's (4.64%); at 20 epochs the two are nearly identical (2.84% vs. 2.72%), so the two regimes are now compared at naturally matched compute reduction without recalibration. Figure 6 visualizes this frontier. In relative terms, joint additive retains 98.41% of the 20-epoch baseline's accuracy (up from 97.19% at 10), while joint coupled retains 91.75% (up from 81.18%).

Table 2 (main text) re-runs the same five regimes for 50 epochs. The additive-coupled gap falls to just 2.21 points at 50 epochs – a smooth, monotonically shrinking sequence ($14.12 \to 6.10 \to 2.21$) rather than a plateau. Joint coupled recovers to 96.08% of baseline accuracy, while joint additive is essentially flat (98.45% vs. 98.41% at 20 and 97.19% at 10). The effective-MAC ordering flips again: coupled was better than additive at 10 epochs, about equal at 20, and now clearly worse at 50 (1.85% vs. 1.18%).

Figure 7 extends this to a full per-layer breakdown of *unstructured* weight sparsity at the same 50-epoch checkpoint, the direct analogue of Figure 9's structured-sparsity view. Reinforcement is essentially flat at the aggregate level by this point: the per-layer weight-sparsity gap averages just $+0.18$ points (range -1.68 to $+1.93$), down from the 10-epoch gap of $+0.66$ – a ceiling effect, since weight-only alone already reaches 93.58% mean sparsity by 50 epochs. Activation-sparsity reinforcement remains positive at every layer (mean $+4.43$, range $+1.18$ to $+8.97$), and the two gap vectors are no longer positively correlated ($r = -0.38$) the way the structured-sparsity gaps were ($r = 0.51$ at 10 epochs, $r = 0.29$ at 50). Joint coupled's per-layer weight sparsity looks qualitatively different from its structured counterpart: every layer reaches substantial unstructured sparsity (56.4–96.7%, no layer below half), suggesting the coupled objective's convergence-speed recovery completes faster at the unstructured granularity.

J.2 ResNet-18/CIFAR-100 at 50 and 100 Epochs

Table 31 re-runs all five Table 9 regimes for 50 epochs.

Table 29: Hybrid additive+coupled objective on ResNet-18/CIFAR-10, 10 epochs, mean ± std over 3 seeds, vs. the pure additive and pure coupled rows from Table 1.

Regime	Accuracy	Weight Sparsity	Activation Sparsity	Effective-MAC Fraction
Joint additive (weight-heavy)	$86.40 \pm 0.40\%$	$86.75 \pm 0.13\%$	$67.57 \pm 2.38\%$	$4.64 \pm 0.12\%$
Hybrid (λ_c=0.2)	$83.82 \pm 1.08\%$	$87.95 \pm 0.12\%$	$73.73 \pm 1.16\%$	$3.39 \pm 0.08\%$
Hybrid (λ_c=0.4)	$80.62 \pm 1.37\%$	$89.24 \pm 0.26\%$	$76.53 \pm 0.88\%$	$2.64 \pm 0.06\%$
Joint coupled ($\lambda_c = 0.8$, alone)	$72.17 \pm 1.59\%$	$71.86 \pm 1.03\%$	$84.30 \pm 1.22\%$	$3.93 \pm 0.27\%$

Table 30: Five-regime comparison on ResNet-18/CIFAR-10, 20 epochs, mean ± std over 3 seeds (same coefficients as Table 1, which uses 10 epochs).

Regime	Accuracy	Weight Sparsity	Activation Sparsity	Effective-MAC Fraction
Baseline	$91.46 \pm 0.18\%$	$4.64 \pm 0.06\%$	$51.32 \pm 0.05\%$	$45.97 \pm 0.04\%$
Weight-only ($\lambda_w = 0.3$)	$90.71 \pm 0.12\%$	$89.52 \pm 0.54\%$	$53.94 \pm 0.39\%$	$4.91 \pm 0.25\%$
Activation-only ($\lambda_a = 0.05$)	$91.03 \pm 0.12\%$	$6.45 \pm 0.22\%$	$68.94 \pm 1.20\%$	$29.13 \pm 1.24\%$
Joint additive (weight-heavy)	$90.01 \pm 0.19\%$	$\mathbf{90.09 \pm 0.25}\%$	$75.71 \pm 0.67\%$	$2.72 \pm 0.07\%$
Joint coupled ($\lambda_c = 0.8$)	$83.91 \pm 1.10\%$	$83.55 \pm 0.32\%$	$\mathbf{80.93 \pm 0.57}\%$	$2.84 \pm 0.07\%$

Joint coupled improves 19.83 accuracy points versus joint additive's 10.41. The additive-coupled gap shrinks from 15.31 points at 10 epochs to 5.89 at 50. Figure 8 visualizes this frontier.

A third checkpoint (Table 32, 100 epochs) shows the gap-closure decelerating rather than continuing at the same rate: $15.31 \rightarrow 5.89 \rightarrow 3.47$ points at 10/50/100 epochs, the second halving much smaller than the first, in contrast to CIFAR-10's closer-to-proportional shrinkage ($14.12 \rightarrow 6.10 \rightarrow 2.21$). Joint coupled's retention keeps climbing ($67.87\% \rightarrow 86.80\% \rightarrow 89.83\%$), but joint additive's dips slightly from 50 to 100 epochs ($95.06\% \rightarrow 94.66\%$).

J.3 MobileNetV2 at 50 Epochs

Table 33 re-runs all five MobileNetV2/CIFAR-10 regimes for 50 epochs.

Joint coupled improves 25.50 points versus joint additive's 8.78. The gap shrinks from 34.29 points at 10 epochs to 17.57 at 50 – more than halved, but not fully closed: joint coupled's seed variance remains high (± 3.55) rather than tightening the way joint additive's does (± 0.23). Table 34 repeats this extension for MobileNetV2/CIFAR-100.

Joint additive improves modestly ($46.22\% \rightarrow 66.49\%$), but joint coupled's recovery is far weaker than in every other cell of this paper ($10.96\% \rightarrow 36.22\%$), leaving the gap at 30.27 points, only a 14.2% relative reduction from 35.26 – the weakest convergence-speed recovery observed anywhere in this paper. Seed variance also fails to tighten ($\pm 6.64 \rightarrow \pm 7.77$), consistent with the same territory as SVHN's chaotic bimodality rather than a smooth, training-length-fixable problem.

J.4 Structured (Channel-Level) Sparsity at 50 Epochs

Table 35 re-runs all five CIFAR-10 structured regimes for 50 epochs.

Joint coupled improves $+35.34$ points versus joint additive's $+4.56$, and its channel sparsity recovers from near-total collapse to $43.96 \pm 2.76\%$ (still below joint additive's 79.87%). The additive-coupled gap shrinks from 37.10 points at 10 epochs to just 6.32 at 50 – an 83.0% relative reduction, closely matching unstructured CIFAR-10's 84.5% over the identical range. We repeat the layer-wise instrumentation at 50 epochs (Figure 9): where the 10-epoch coupled model reached exactly 0% channel sparsity in 18 of 20 layers, the 50-epoch model reaches 0% in only *one* layer, with the other 19 all showing some structured sparsity (mean 35.16%, up from 1.43%). The per-layer channel/activation gap correlation weakens from $r = 0.51$ at 10 epochs to $r = 0.29$ at 50, consistent with a ceiling effect as weight-only alone reaches higher sparsity in many layers by 50 epochs.

Table 36 repeats the same extension on CIFAR-100.

Joint coupled improves from $21.74 \pm 8.57\%$ to $55.98 \pm 1.00\%$ accuracy, and channel sparsity recovers from 0.39% to $32.22 \pm 1.50\%$ – still well below joint additive's 70.77%, and proportionally further

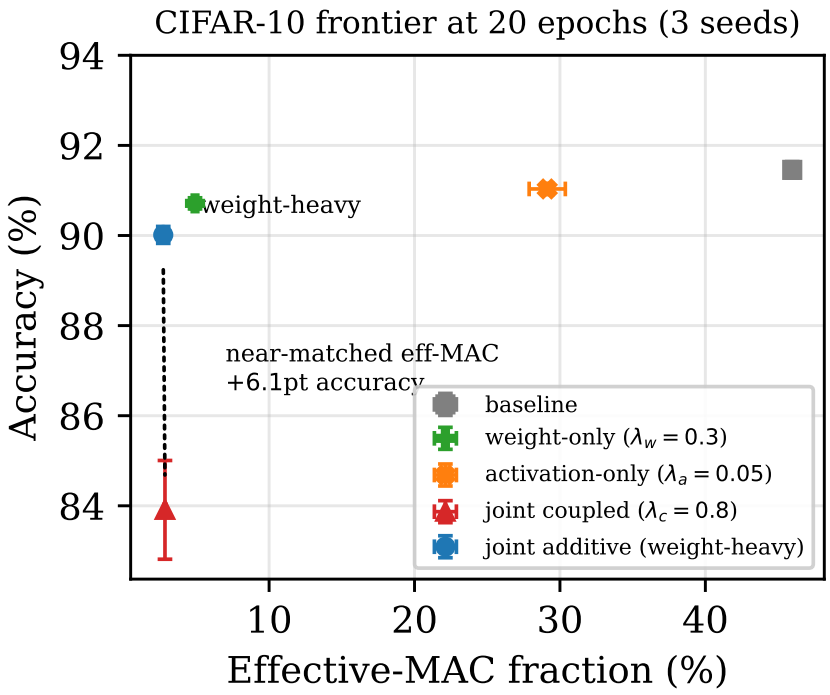


Figure 6: CIFAR-10 frontier at 20 epochs (3 seeds), cf. the 10-epoch frontier in Figure 1. Only the sweep-selected primary configuration was re-run at 20 epochs, not the other three additive candidates from Table 7.

Table 31: Five-regime comparison on ResNet-18/CIFAR-100, 50 epochs, mean ± std over 3 seeds.

Regime	Accuracy	Weight Sparsity	Activation Sparsity	Effective-MAC Fraction
Baseline	$71.39 \pm 0.50\%$	$4.85 \pm 0.16\%$	$55.26 \pm 0.14\%$	$42.37 \pm 0.23\%$
Weight-only ($\lambda_w = 0.3$)	$69.92 \pm 0.48\%$	$91.76 \pm 0.31\%$	$62.36 \pm 0.84\%$	$3.19 \pm 0.16\%$
Activation-only ($\lambda_a = 0.05$)	$69.44 \pm 0.49\%$	$5.25 \pm 0.42\%$	$73.05 \pm 0.79\%$	$25.24 \pm 0.78\%$
Joint additive (weight-heavy)	$67.86 \pm 0.52\%$	$\mathbf{91.90 \pm 0.50}\%$	$77.81 \pm 1.03\%$	$2.06 \pm 0.18\%$
Joint coupled ($\lambda_c = 0.6$)	$61.97 \pm 0.87\%$	$87.15 \pm 0.20\%$	$\mathbf{77.97 \pm 0.55}\%$	$\mathbf{2.65 \pm 0.02}\%$

behind than CIFAR-10's 50-epoch recovery. The additive-coupled gap shrinks from 36.50 to 11.46 points (a 68.6% relative reduction, smaller than CIFAR-10's 83.0%), consistent with the pattern that CIFAR-100 recovers from the coupled formulation's early disadvantage less completely than CIFAR-10 over the same budget. Seed variance also tightens dramatically ($\pm 8.57 \rightarrow \pm 1.00$), unlike the elevated variance that persists in other structured or bimodal cases – here, more training resolves both the accuracy gap and the instability together.

K Effective-MAC Validation and Covariance Decomposition

We instrument the real forward pass: for every `Conv2d` layer we extract the literal input patches via im2col (`F.unfold`) and compute, via a matrix multiply of the weight and activation nonzero-masks, the exact count of (weight, activation) products with both operands nonzero – the true AND-condition a MAC requires, with no independence assumption and no depth-pairing approximation. We retrain the five Table 1 regimes (same coefficients, 10 epochs, single seed) and instrument each on a held-out batch. Table 37 compares the proxy against an unweighted layer-mean and a MAC-weighted global fraction (the number that determines actual compute savings).

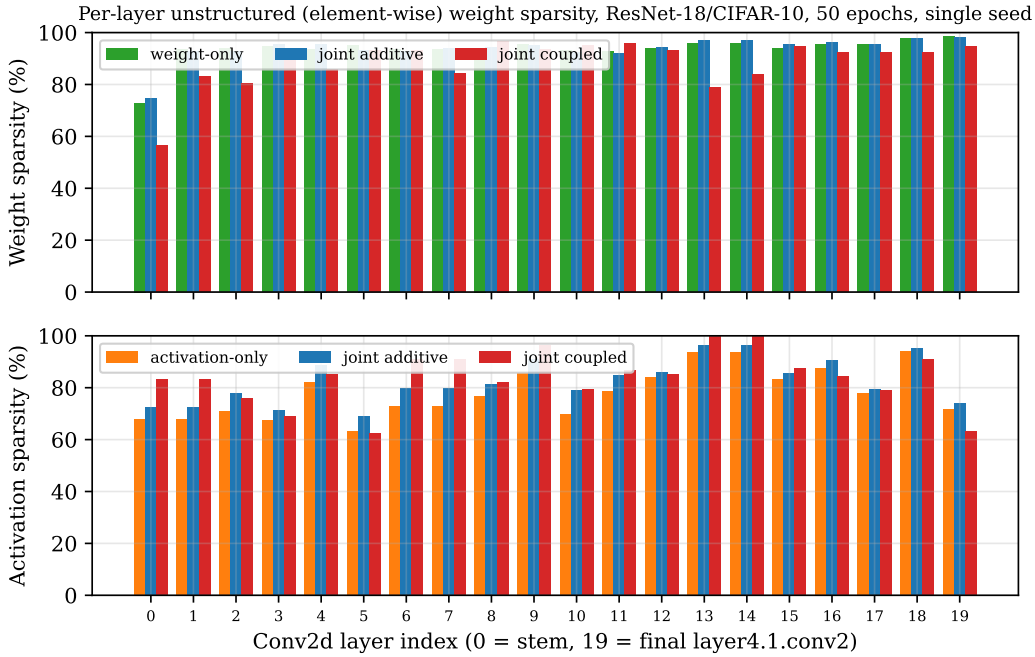


Figure 7: Per-layer unstructured (element-wise) weight sparsity (top) and activation sparsity (bottom) across all 20 `Conv2d` layers of ResNet-18/CIFAR-10, single seed, 50 epochs. Unlike the structured case (Figure 9), joint coupled (red) reaches substantial weight sparsity at every layer, and reinforcement is small and depth-independent by this checkpoint.

Table 32: Five-regime comparison on ResNet-18/CIFAR-100, 100 epochs, mean $\pm$ std over 3 seeds (same coefficients as Table 9 and Table 31).

Regime	Accuracy	Weight Sparsity	Activation Sparsity	Effective-MAC Fraction
Baseline	$71.90 \pm 0.45\%$	$4.86 \pm 0.15\%$	$58.00 \pm 0.15\%$	$39.85 \pm 0.23\%$
Weight-only ($\lambda_w = 0.3$)	$70.61 \pm 0.30\%$	$95.97 \pm 0.15\%$	$66.72 \pm 0.67\%$	$1.38 \pm 0.06\%$
Activation-only ($\lambda_a = 0.05$)	$69.48 \pm 0.69\%$	$5.21 \pm 0.28\%$	$77.85 \pm 0.64\%$	$20.70 \pm 0.64\%$
Joint additive (weight-heavy)	$68.06 \pm 0.24\%$	$\mathbf{95.95 \pm 0.24}\%$	$81.77 \pm 0.28\%$	$\mathbf{0.84 \pm 0.05}\%$
Joint coupled ($\lambda_c = 0.6$)	$64.59 \pm 1.58\%$	$93.04 \pm 0.38\%$	$\mathbf{79.04 \pm 0.69}\%$	$1.34 \pm 0.12\%$

The proxy and the MAC-weighted exact ground truth diverge substantially for the dense baseline (48.15% vs. 41.02%) but converge closely for every regularized regime, almost exactly for the two sparsest (joint additive: 0.64 points; joint coupled: 0.01 points) – reassuring for this paper's central compute-savings claims, though the independence assumption is measurably imperfect in general. The unweighted layer-mean tells a less flattering story, diverging *more* as regularization gets more aggressive (1.16 points at baseline, widening to 3.43 for joint coupled), tracing to non-uniform layer sizes: aggregating C_l by simple layer-mean vs. by MAC-weighted mean can disagree when the layers a regularizer sparsifies most are not the layers carrying the most real MAC volume.

Measuring the covariance term directly (Proposition 4). The same forward-pass instrumentation yields the two marginals $P(X{=}1)$, $P(Y{=}1)$ directly, so we compute $\mathrm{Cov}(X, Y)$ as exact-minus-naive on the identical five retrained regimes.

The covariance is positive in every regularized regime and essentially zero at baseline ($+0.00$, range $[-0.01, +0.04]$ across layers) – consistent with Proposition 4's prediction that unregularized weight and activation masks carry no shared structure. Regularizing a single operand leaves the covariance small ($+0.07$–0.08pp). Joint additive is qualitatively different: its covariance rises to $+0.67$pp, an order of magnitude larger and 11.4% relative to its own naive term – regularizing both operands simultaneously induces a real statistical association between which weights and which activations

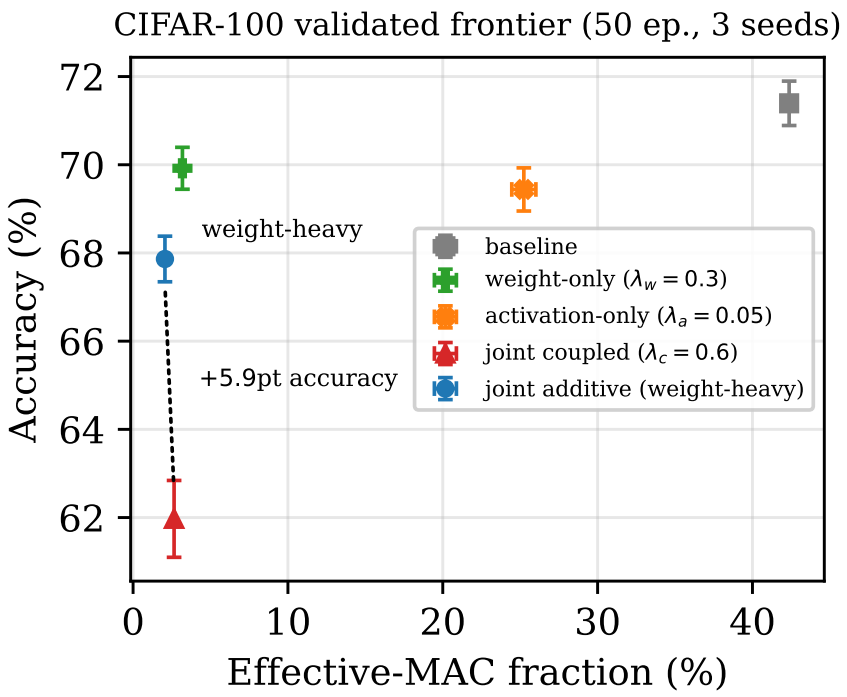


Figure 8: CIFAR-100 validated frontier at 50 epochs (3 seeds), cf. the 10-epoch numbers in Table 9. Joint coupled remains at $\lambda_c = 0.6$, calibrated to match weight-heavy's 10-epoch effective-MAC fraction.

Table 33: Five-regime comparison on MobileNetV2/CIFAR-10, 50 epochs, mean ± std over 3 seeds, same coefficients as Table 15.

Regime	Accuracy	Weight Sparsity	Activation Sparsity	Effective-MAC Fraction
Baseline	$90.56 \pm 0.12\%$	$2.57 \pm 0.05\%$	$50.13 \pm 0.17\%$	$48.96 \pm 0.11\%$
Weight-only ($\lambda_w = 0.3$)	$89.82 \pm 0.31\%$	$90.91 \pm 0.12\%$	$59.70 \pm 0.14\%$	$4.12 \pm 0.07\%$
Activation-only ($\lambda_a = 0.05$)	$89.63 \pm 0.09\%$	$3.03 \pm 0.02\%$	$78.21 \pm 0.44\%$	$22.08 \pm 0.48\%$
Joint additive (weight-heavy)	$89.10 \pm 0.23\%$	$\mathbf{91.53 \pm 0.18}\%$	$84.34 \pm 0.31\%$	$1.73 \pm 0.08\%$
Joint coupled ($\lambda_c = 0.8$, unrecalibrated)	$71.53 \pm 3.55\%$	$63.19 \pm 1.18\%$	$\mathbf{95.91 \pm 0.35}\%$	$\mathbf{1.58 \pm 0.13}\%$

survive, not merely two independent marginal reductions a product of marginals would already capture. Joint coupled's covariance (+0.36pp) sits between the single-axis and joint-additive regimes.

L Statistical Significance

Tables 1–16 and Table 5 report means and standard deviations (five seeds for Table 1, three seeds elsewhere), but a standard deviation alone does not establish statistical distinguishability from noise. We run a two-sided Welch's t-test comparing each solo regime's sparsity to joint additive's (14 tests, Table 39, Benjamini-Hochberg corrected among themselves), and separately for the joint-additive-vs-joint-coupled accuracy gap in each of the five settings where both regimes are well-defined (5 tests, corrected among themselves) – two logically distinct families of hypotheses, each tests a different scientific claim (reinforcement vs. the coupled objective's underperformance), so we correct each separately rather than pooling them, a more standard FDR practice than the single 17-test pool used in an earlier draft of this analysis. Structured (channel-level) sparsity's own marginal-sparsity tests (Section 4.3: weight $p = 0.0006$, activation $p = 0.0067$) form a third family, since granularity is a distinct scientific question from either of the first two rather than another architecture/task/domain setting within the unstructured comparison; both survive Benjamini-Hochberg correction within this

Table 34: Five-regime comparison on MobileNetV2/CIFAR-100, 50 epochs, mean ± std over 3 seeds, same coefficients as Table 16.

Regime	Accuracy	Weight Sparsity	Activation Sparsity	Effective-MAC Fraction
Baseline	$67.10 \pm 0.42\%$	$2.66 \pm 0.05\%$	$49.41 \pm 0.04\%$	$49.70 \pm 0.05\%$
Weight-only ($\lambda_w = 0.3$)	$67.55 \pm 0.17\%$	$88.49 \pm 0.23\%$	$57.57 \pm 0.66\%$	$5.36 \pm 0.19\%$
Activation-only ($\lambda_a = 0.05$)	$65.90 \pm 0.19\%$	$2.92 \pm 0.09\%$	$72.56 \pm 0.72\%$	$27.64 \pm 0.74\%$
Joint additive (weight-heavy)	$66.49 \pm 0.16\%$	$\mathbf{88.79 \pm 0.26\%}$	$79.88 \pm 0.59\%$	$2.87 \pm 0.12\%$
Joint coupled ($\lambda_c = 0.6$)	$36.22 \pm 7.77\%$	$60.34 \pm 0.96\%$	$\mathbf{95.70 \pm 0.78\%}$	$\mathbf{1.53 \pm 0.33\%}$

Table 35: Five-regime comparison with structured (channel-level) weight sparsity on ResNet-18/CIFAR-10, 50 epochs, mean ± std over 3 seeds, same coefficients as Table 6.

Regime	Accuracy	Channel Sparsity	Activation Sparsity	Effective-MAC Fraction
Baseline	$93.11 \pm 0.09\%$	$0.00 \pm 0.00\%$	$56.11 \pm 0.10\%$	$43.48 \pm 0.13\%$
Weight-only ($\lambda_w = 0.3$)	$92.39 \pm 0.14\%$	$75.30 \pm 1.22\%$	$74.66 \pm 0.36\%$	$7.21 \pm 0.62\%$
Activation-only ($\lambda_a = 0.05$)	$92.55 \pm 0.18\%$	$0.00 \pm 0.00\%$	$77.61 \pm 0.39\%$	$22.52 \pm 0.43\%$
Joint additive (weight-heavy)	$91.22 \pm 0.22\%$	$\mathbf{79.87 \pm 0.17\%}$	$85.52 \pm 0.18\%$	$3.81 \pm 0.21\%$
Joint coupled ($\lambda_c = 0.8$, unrecalibrated)	$84.90 \pm 0.81\%$	$43.96 \pm 2.76\%$	$\mathbf{88.73 \pm 0.26\%}$	$\mathbf{5.23 \pm 0.34\%}$

two-test family ($p_{\text{BH}} = 0.0012$ and 0.0067 respectively). WikiText-2's own joint-additive-vs-joint-coupled validation-loss gap (Appendix P: $t = 6.49$, $p = 0.0038$) is reported uncorrected, as a single comparison outside the paper's main significance families, since a validation-loss gap on a language model is not commensurable with the accuracy gap the five-test family above measures.

Six of the fourteen marginal-sparsity comparisons now survive correction: weight-sparsity reinforcement on MNIST, on MobileNetV2 for both datasets, and – newly – on WikiText-2's word-level language model; activation-sparsity reinforcement on SVHN and, at the upgraded $n = 5$, on CIFAR-10/ResNet-18, this paper's own headline comparison. CIFAR-10's weight-sparsity gap also strengthens substantially (raw $p = 0.033$, individually significant before correction) but does not quite clear the corrected threshold ($p_{\text{BH}} = 0.066$). The remaining eight do not clear significance at their sample size, a genuine power limitation for those specific settings rather than evidence against the effect: twelve of fourteen comparisons still point in the reinforcing direction, the two exceptions being activation sparsity on MNIST and, newly, activation sparsity on WikiText-2. We report a one-sided sign test under a null of random sign ($p \approx 0.006$) for completeness, but read it only as a description of directional consistency, not a formal test: the fourteen comparisons share regularizer and implementation and mostly share architecture family, so they are not independent replications in the sense the test assumes. The joint-additive-vs-joint-coupled accuracy gap is a different story: significant after its own correction in every setting tested (raw $p < 0.0001$ for CIFAR-10 at $n = 5$; ResNet-18/CIFAR-100, MNIST/LeNet, MobileNetV2/CIFAR-10, and MobileNetV2/CIFAR-100 all previously significant at $n = 3$ and unaffected by this update) – the more statistically robust of this paper's two headline findings, and now joined both by activation-sparsity reinforcement on CIFAR-10 and by WikiText-2's own additive-vs-coupled validation-loss gap (reported separately above) as individually-significant results in their own right.

M Adversarial Robustness (Exploratory)

Beyond compute efficiency, does joint sparsity change robustness to adversarial perturbations? We test this as a minimally-scoped exploratory check, not with the full multi-seed treatment given to the paper's central claims. On LeNet/MNIST, single seed, original un-tuned coefficients, we attack each trained regime with untargeted FGSM (five L_∞ budgets) and 10-step PGD (Table 40; $\epsilon = 0.487$ is a raw-pixel budget of 0.15).

Activation sparsity, not weight sparsity, correlates with the robustness gain ($r = 0.91$ with PGD accuracy vs. $r = 0.13$ for weight sparsity), and joint coupled – this paper's consistently weaker regime on accuracy and compute – is the most robust regime tested at every budget, consistent with activation sparsity acting as an implicit thresholding operation on the perturbation, similar in spirit to feature-squeezing defenses. We treat this as suggestive rather than validated: single seed, single architecture, no adaptive attacks tested.

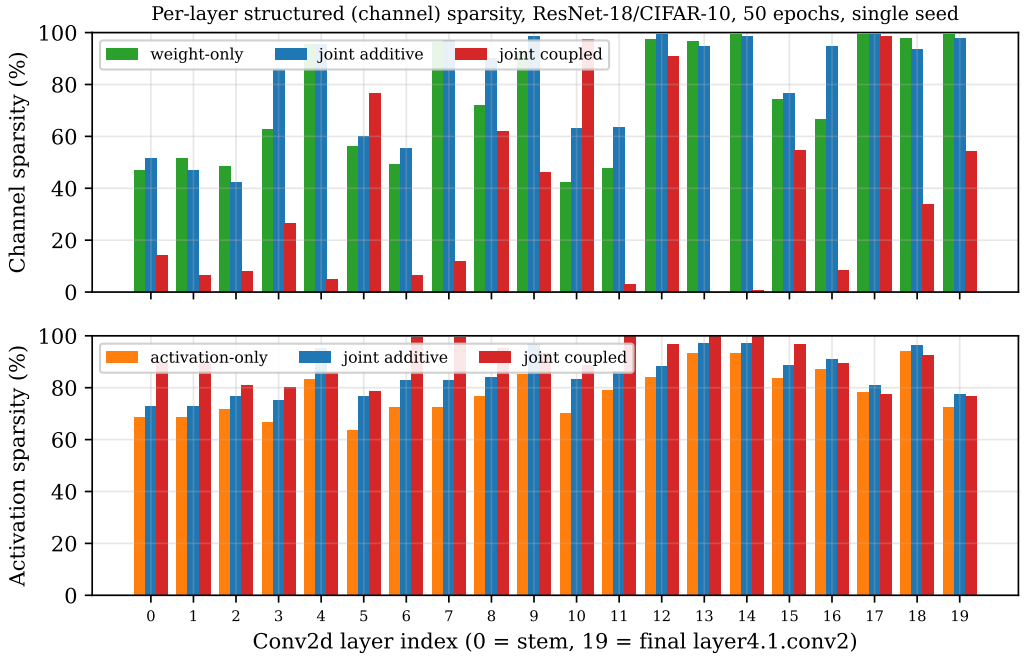


Figure 9: Per-layer channel sparsity (top) and activation sparsity (bottom) at 50 epochs, cf. the 10-epoch version in Figure 5. Joint coupled (red) now shows substantial channel sparsity at nearly every layer, rather than only two.

Table 36: Five-regime comparison with structured (channel-level) weight sparsity on ResNet-18/CIFAR-100, 50 epochs, mean ± std over 3 seeds, same coefficients as Table 20.

Regime	Accuracy	Channel Sparsity	Activation Sparsity	Effective-MAC Fraction
Baseline	$71.04 \pm 0.17\%$	$0.00 \pm 0.00\%$	$55.32 \pm 0.17\%$	$44.31 \pm 0.16\%$
Weight-only ($\lambda_w = 0.3$)	$69.39 \pm 1.01\%$	$66.26 \pm 0.44\%$	$71.44 \pm 1.43\%$	$11.40 \pm 0.05\%$
Activation-only ($\lambda_a = 0.05$)	$69.31 \pm 0.59\%$	$0.00 \pm 0.00\%$	$72.89 \pm 0.74\%$	$26.80 \pm 0.78\%$
Joint additive (weight-heavy)	$67.44 \pm 0.60\%$	$\mathbf{70.77 \pm 0.96}\%$	$81.07 \pm 0.19\%$	$7.51 \pm 0.07\%$
Joint coupled ($\lambda_c = 0.6$, unrecalibrated)	$55.98 \pm 1.00\%$	$32.22 \pm 1.50\%$	$\mathbf{86.08 \pm 0.39}\%$	$\mathbf{8.45 \pm 0.50}\%$

This does not replicate on ResNet-18/CIFAR-10. Repeating the identical protocol with CIFAR-10's sweep-selected coefficients at a standard $8/255$ raw-pixel budget ($\epsilon = 0.156$; Table 41), PGD collapses every regime to exactly 0.00% – this attack strength defeats any naturally-trained (non-adversarially-trained) CIFAR-10 model regardless of sparsity regime – and under FGSM the relationship *reverses*: activation sparsity correlates *negatively* with FGSM accuracy ($r = -0.62$), and so does weight sparsity ($r = -0.64$), while clean accuracy correlates strongly positively ($r = 0.88$).

Joint coupled – the most robust regime on MNIST – is here the *least* robust under FGSM (1.10%), but it is also far less accurate on clean inputs (72.91% vs. 85.93–88.65% for the other four). MNIST's five regimes span under one point of clean accuracy, so an independent sparsity effect is plausible there; CIFAR-10's span nearly sixteen points, making a weaker classifier's weaker accuracy floor under attack the more parsimonious explanation than an independent robustness relationship running in the opposite direction. We do not treat this as evidence against the MNIST finding specifically, but it substantially weakens any claim that joint sparsity confers a general robustness benefit: at the one other setting tested, the effect did not transfer.

N A Vision Transformer Probe (Preliminary)

Every architecture validated elsewhere in this paper is a ReLU (or ReLU6) convolutional network. We test whether this paper's two central findings – marginal reinforcement, and the coupled objective's

Table 37: Effective-MAC proxy vs. exact instrumentation on ResNet-18/CIFAR-10, 10 epochs, single seed (same coefficients as Table 1). "Exact (MAC-weighted)" is the ground-truth compute-savings number; "Exact (layer-mean)" matches the proxy's own per-layer averaging.

Regime	Accuracy	Proxy (Independence)	Exact (Layer-Mean)	Exact (MAC-Weighted)
Baseline	88.42%	48.15%	46.99%	41.02%
Weight-only ($\lambda_w = 0.3$)	87.14%	7.54%	8.48%	5.72%
Activation-only ($\lambda_a = 0.05$)	87.46%	31.92%	33.24%	27.58%
Joint additive (weight-heavy)	85.83%	4.58%	6.60%	3.94%
Joint coupled ($\lambda_c = 0.8$)	72.48%	3.64%	7.06%	**3.63%**

Table 38: Empirical covariance decomposition (Proposition 4) on ResNet-18/CIFAR-10, 10 epochs, single seed, same regimes and coefficients as Table 37. All values are unweighted layer-means; $\mathrm{Cov}(X, Y)$ is exact minus naive, in percentage points.

Regime	$P(X{=}1)$	$P(Y{=}1)$	**Naive** $P(X)P(Y)$	**Exact** C_l^{exact}	$\mathrm{Cov}(X, Y)$
Baseline	95.50%	49.19%	47.17%	47.18%	+0.00
Weight-only ($\lambda_w = 0.3$)	14.78%	48.66%	8.36%	8.44%	+0.08
Activation-only ($\lambda_a = 0.05$)	93.69%	34.86%	33.09%	33.16%	+0.07
Joint additive (weight-heavy)	13.60%	30.43%	5.88%	6.56%	**+0.67**
Joint coupled ($\lambda_c = 0.8$)	29.23%	18.20%	7.13%	7.49%	+0.36

lopsided weight-vs-activation failure signature – say anything about attention architectures with a small, explicitly preliminary probe: a 0.5M-parameter Vision Transformer on CIFAR-10 (patch size 4, embedding dimension 128, 4 pre-norm transformer blocks, 4 attention heads), using the same five-regime protocol, single seed, coefficients chosen only to avoid outright collapse from a 5-epoch screen (Section C's methodology, not a full sweep). We test two MLP nonlinearities: GELU, the standard choice for real Transformers, and ReLU, which sacrifices architectural realism but keeps this paper's exact-zero sparsity metric directly comparable to every other table in the paper.

The exact-zero activation-sparsity metric is degenerate for GELU. Every regime in this paper's other experiments applies ReLU or ReLU6, both of which clip negative inputs to a literal 0.0, so counting exact zeros (Section 3.1) is a meaningful sparsity measure. GELU has no such clipping point and essentially never outputs exactly 0.0: exact-zero activation sparsity stays below 0.5% in every regime tested here, including joint additive at its full coefficient (Table 42) – not because regularization has no effect, but because the metric itself cannot see it. We report the continuous per-element-normalized squared-Hoyer measure $H(A)$ directly instead, which is well-defined and responsive regardless of the nonlinearity, since it is the quantity the regularizer actually optimizes.

Marginal reinforcement replicates under the corrected metric. Joint additive's weight sparsity (65.31%) exceeds weight-only's (63.56%) by +1.75 points, and joint additive's $H(A)$ (0.0697) is *lower* (more concentrated) than activation-only's (0.0961) – both marginal axes reinforce, the same qualitative pattern reported throughout this paper's CNN experiments, on a single seed with no significance testing.

The coupled objective's lopsided failure signature also replicates. Joint coupled reaches far less weight sparsity than additive (6.77% vs. 65.31%) while pushing $H(A)$ to the lowest value of any regime tested (0.0244, lower even than activation-only's 0.0961) – the identical "trades weight sparsity for activation concentration" asymmetry Table 1 reports for ResNet-18/CIFAR-10, now appearing on a completely different architecture family (attention plus GELU MLPs rather than convolution plus ReLU).

The naive effective-MAC formula collapses to tracking weight sparsity alone here. Because exact-zero activation sparsity never exceeds 0.5%, the naive-independence effective-MAC estimate (Section 3.1) reduces almost exactly to $1 - (\text{weight sparsity})$ in every regime (e.g. joint additive: $1 - 65.31\% = 34.69\%$ vs. the reported 34.58%) – it is not a meaningful compute-reduction estimate for this architecture as defined, since it credits no reduction at all for GELU's real (if not exactly-zero) sparsification.

Table 39: Welch's t-test for marginal-sparsity reinforcement (solo regime vs. joint additive) across every five-regime table in the paper ($n = 5$ seeds for CIFAR-10/ResNet-18, $n = 3$ elsewhere). p_{BH} is Benjamini-Hochberg-corrected across these 14 tests. Starred rows survive at $\alpha = 0.05$ after correction.

Setting	**Comparison**	**Gap**	t	p	$p_{\textbf{BH}}$
CIFAR-10/ResNet-18 (n=5)	weight sparsity	$+0.68$	-3.06	0.033	0.066
CIFAR-10/ResNet-18 (n=5)	activation sparsity	$+4.63$	-3.33	0.011	$\mathbf{0.026}$*
CIFAR-100/ResNet-18	weight sparsity	$+1.37$	-3.02	0.087	0.121
CIFAR-100/ResNet-18	activation sparsity	$+5.95$	-2.01	0.123	0.144
MNIST/LeNet	weight sparsity	$+9.68$	-9.17	0.002	$\mathbf{0.009}$*
MNIST/LeNet	activation sparsity	-2.91	2.35	0.095	0.121
SVHN/ResNet-18	weight sparsity	$+0.44$	-3.64	0.052	0.091
SVHN/ResNet-18	activation sparsity	$+6.70$	-31.77	0.001	$\mathbf{0.007}$*
MobileNetV2/CIFAR-10	weight sparsity	$+1.30$	-6.03	0.004	$\mathbf{0.011}$*
MobileNetV2/CIFAR-10	activation sparsity	$+0.39$	-0.76	0.520	0.520
MobileNetV2/CIFAR-100	weight sparsity	$+2.59$	-11.61	0.0004	$\mathbf{0.006}$*
MobileNetV2/CIFAR-100	activation sparsity	$+3.21$	-3.15	0.072	0.112
WikiText-2/CharTransformer	weight sparsity	$+2.43$	-12.72	0.0036	$\mathbf{0.011}$*
WikiText-2/CharTransformer	activation sparsity	-0.63	1.47	0.270	0.290

Table 40: Adversarial robustness across five regimes, LeNet/MNIST, single seed, original coefficients. FGSM/PGD ϵ values are normalized-input units; $\epsilon = 0.487$ is a raw-pixel budget of 0.15.

Regime	**Clean Acc.**	**Weight Sp.**	**Act. Sp.**	**FGSM** (ϵ=0.649)	**PGD** (ϵ=0.487)
Baseline	99.14%	3.18%	51.93%	23.40%	15.58%
Weight-only ($\lambda_w = 0.5$)	98.39%	74.72%	44.60%	36.69%	13.78%
Activation-only ($\lambda_a = 0.2$)	99.03%	3.94%	76.85%	41.22%	21.24%
Joint additive	98.49%	84.71%	72.16%	38.24%	21.18%
Joint coupled ($\lambda_c = 0.5$)	98.95%	60.29%	**77.75%**	**53.03%**	**27.62%**

A ReLU variant restores a directly comparable metric, and tells a more mixed story. We repeat the identical protocol with the MLP nonlinearity swapped to ReLU (same coefficients, independently re-screened to confirm they still avoid collapse; Table 43). Exact-zero activation sparsity is no longer degenerate (74.18% at baseline, comparable in magnitude to this paper's CNN baselines), confirming the GELU result above was a metric artifact rather than a real difference in what regularization achieves.

The picture under ReLU is more mixed than under GELU, not simply confirmatory. Weight-sparsity reinforcement still holds (joint additive 64.23% vs. weight-only 63.51%, +0.72 points), and $H(A)$ still reinforces (joint additive 0.0352 vs. activation-only 0.0387) – but the exact-zero activation-sparsity metric now shows the *opposite* of reinforcement, if only slightly (joint additive 88.73% vs. activation-only's 89.50%, −0.77 points, on one seed). More consequentially, the aggregate effective-MAC comparison – computed properly here, per-layer paired sparsities rather than global means, matching this paper's standard convention – shows joint additive's actual effective-MAC (5.85%) is slightly *worse* (higher) than the naive independence estimate from the two solo regimes (5.66%), meaning aggregate effective-compute reinforcement fails here, joining MobileNetV2/CIFAR-10 (Section 4.3) as a second documented exception to that particular claim. The coupled objective's lopsided signature replicates a third time regardless: joint coupled reaches far less weight sparsity than additive (7.78% vs. 64.23%) while its activation sparsity and $H(A)$ are the most extreme of any regime tested (96.30%, $H(A) = 0.0094$).

We treat this entire probe as suggestive rather than validated: one seed per variant, one small model (0.5M parameters, 4 layers), coefficients screened only to avoid collapse rather than swept for a Pareto-optimal operating point, and, at the time of this probe, no test of whether any of this holds on a non-vision task (Appendix O takes a first, equally preliminary, look at exactly that). Taken together, the two variants suggest the coupled objective's lopsided failure signature is the more robust finding

Table 41: Adversarial robustness across five regimes, ResNet-18/CIFAR-10, single seed, sweep-selected coefficients (same as Table 1). FGSM/PGD $\epsilon = 0.156$ corresponds to a raw-pixel budget of $8/255$.

Regime	Clean Acc.	Weight Sp.	Act. Sp.	FGSM (ϵ=0.156)	PGD (ϵ=0.156)
Baseline	88.65%	4.48%	48.62%	3.90%	0.00%
Weight-only ($\lambda_w = 0.3$)	87.74%	85.27%	49.29%	3.25%	0.00%
Activation-only ($\lambda_a = 0.05$)	87.22%	6.04%	65.66%	**4.89%**	0.00%
Joint additive	85.93%	86.53%	70.90%	3.24%	0.00%
Joint coupled ($\lambda_c = 0.8$)	72.91%	71.65%	85.32%	1.10%	0.00%

Table 42: Five-regime comparison on a small ViT/CIFAR-10 with GELU MLPs, 10 epochs, single seed, coefficients screened only to avoid collapse. "Act. Sp. (exact-0)" is the same metric used for the ReLU networks elsewhere in this paper; $H(A)$ is the continuous Hoyer measure the regularizer directly optimizes. "Naive Eff.-MAC" reuses this paper's standard formula but is not a meaningful compute estimate here (see text).

Regime	Accuracy	Weight Sp.	Act. Sp. (exact-0)	$H(A)$	Naive Eff.-MAC
Baseline	71.06%	2.73%	0.01%	0.3336	97.27%
Weight-only ($\lambda_w = 0.1$)	68.87%	63.56%	0.19%	0.2807	36.39%
Activation-only ($\lambda_a = 0.05$)	70.70%	3.92%	0.12%	0.0961	95.96%
Joint additive	67.65%	**65.31%**	0.41%	**0.0697**	34.58%
Joint coupled ($\lambda_c = 0.1$)	66.49%	6.77%	0.32%	0.0244	92.93%

of the two across architecture families tested so far, while marginal and aggregate reinforcement are, consistent with this paper's own "conditional, not uniform" framing (Section 5), sensitive to exactly which nonlinearity and which specific metric is used to look for them.

O A Character-Level Language Model Probe (Preliminary)

Every setting tested elsewhere in this paper, including the ViT probe above, is a vision task. We take a first, equally preliminary look at a non-vision domain: a 0.56M-parameter decoder-only (causal) Transformer, reusing the same block architecture as Appendix N with causal self-attention masking and ReLU MLPs (avoiding the GELU exact-zero degeneracy documented there directly, rather than working around it a second time), trained for character-level language modeling on Tiny Shakespeare (dim 128, 4 layers, 4 heads, context length 128, vocabulary size 65). We use the same five-regime protocol, single seed, coefficients screened at 500 steps only to avoid collapse, then run for 3000 training steps (≈ 190 passes over the corpus at this batch size and context length). The token embedding is excluded from the weight set regularized and measured, since it performs no multiply-accumulates (a lookup, not a matmul) – only the four Linear weights per block plus the output head are regularized and counted toward effective-MAC, an architectural distinction that does not arise in any of this paper's vision models.

Marginal reinforcement splits across the two operands, the same pattern a companion ReLU-ViT probe also shows. Joint additive's weight sparsity (60.69%) exceeds weight-only's (58.84%) by $+1.85$ points – reinforcing, consistent with every vision setting. Its activation sparsity (78.95%), however, is *lower* than activation-only's alone (81.27%), a -2.32-point reversal – on the identical operand, though larger in magnitude, as Appendix N's ReLU-ViT probe's own -0.77-point reversal. Only the GELU-ViT variant (Appendix N), whose degenerate exact-zero metric forces using the continuous $H(A)$ measure instead, shows both operands reinforcing. Of the four logically possible (weight reinforces or not) $\times$ (activation reinforces or not) combinations, only two have actually been observed across these three non-CNN probes – both operands reinforcing (GELU-ViT) and weight reinforcing while activation reverses (ReLU-ViT, this probe) – so the pattern may track the exact-zero measurement convention as much as it tracks domain or architecture; we do not have enough non-CNN settings to distinguish these explanations.

Aggregate effective-compute reinforcement fails here too. The naive independence estimate from the two solo regimes' global-mean sparsities is $(100 - 58.84\%) \times (100 - 81.27\%) \approx 7.71\%$; joint

Table 43: Five-regime comparison on the identical small ViT/CIFAR-10, MLP nonlinearity swapped to ReLU, 10 epochs, single seed, coefficients re-screened for this variant (Table 42's values still apply). Naive Eff.-MAC here uses each regime's own per-layer paired sparsities, matching this paper's standard convention, since the exact-zero metric is no longer degenerate.

Regime	Accuracy	Weight Sp.	Act. Sp. (exact-0)	$H(A)$	Eff.-MAC
Baseline	69.66%	2.72%	74.18%	0.1478	25.83%
Weight-only ($\lambda_w = 0.1$)	67.88%	63.51%	70.94%	0.1204	11.45%
Activation-only ($\lambda_a = 0.05$)	70.41%	2.91%	**89.50%**	**0.0387**	11.13%
Joint additive	68.21%	**64.23%**	88.73%	0.0352	5.85%
Joint coupled ($\lambda_c = 0.1$)	68.71%	7.78%	96.30%	0.0094	3.91%

Table 44: Five-regime comparison on a character-level Transformer LM, Tiny Shakespeare, 3000 steps, single seed, coefficients screened only to avoid collapse. Lower validation loss and perplexity are better (unlike every other table in this paper, where accuracy is reported and higher is better).

Regime	Val. Loss	Perplexity	Weight Sp.	Act. Sp.	Eff.-MAC
Baseline	1.8161	6.15	2.36%	54.57%	44.23%
Weight-only ($\lambda_w = 0.1$)	1.8699	6.49	58.84%	53.04%	19.30%
Activation-only ($\lambda_a = 0.05$)	1.8096	6.11	2.62%	81.27%	18.32%
Joint additive	1.8571	6.40	**60.69%**	78.95%	8.60%
Joint coupled ($\lambda_c = 0.1$)	**1.7821**	**5.94**	4.02%	**92.09%**	**7.41%**

additive's actual effective-MAC (8.60%) is 11.6% relatively *worse*, a third documented exception alongside MobileNetV2/CIFAR-10 and the ReLU-ViT probe.

The most notable result is the coupled objective's performance, not either reinforcement axis. Joint coupled reaches the *best* validation loss and perplexity of any regime tested here, including baseline (1.7821 vs. 1.8161), while also reaching the lowest effective-MAC fraction of any regime (7.41%) – the opposite of every unstructured vision result in this paper, where the coupled objective trades substantial accuracy for its compute reduction (Table 1: 14.12 points at matched calibration). This makes character-level language modeling only the second setting in this paper, after MNIST/LeNet (Section 4.3), where the coupled formulation does not underperform additive – extending that anomaly from a single architecture/task/calibration coincidence to a genuinely different axis (domain). Activation-only alone also slightly *improves* on baseline (1.8096 vs. 1.8161), consistent with mild sparsity regularization acting as a generalization aid on a corpus this small, an effect also glimpsed (more weakly) in the ReLU-ViT probe's activation-only regime.

We treat this probe as even more preliminary than the ViT probe above: one seed, one tiny model and corpus, a single screened-not-swept coefficient per regime, and no significance testing of any kind. The coupled objective's strong showing here is the single most direct challenge to this paper's own headline finding that the coupled objective underperforms substantially, and is exactly the kind of result that would most benefit from further seeds and a larger corpus before treating it as more than a preliminary flag. Appendix P takes exactly that further step: the same question, a real word-level benchmark an order of magnitude larger, and a proper n=3 multi-seed protocol.

P A Validated Word-Level Language-Model Test: WikiText-2

The character-level probe above is single-seed and uses a tiny, non-standard corpus; we extend it to a standard, larger benchmark with the same multi-seed protocol used for every other generalization setting in this paper. We reuse the identical decoder-only Transformer architecture (ReLU MLPs, causal self-attention, 4 layers, 4 heads, dim 128, context length 128) unchanged, only swapping character-level tokenization for word-level tokenization on WikiText-2 (Merity et al., 2017): 2.09M training tokens (roughly 19× Tiny Shakespeare's corpus) and a real 33,278-word vocabulary in place of 65 characters. As before, the token embedding is excluded from the regularized/measured weight set (a lookup, not a matmul); the output head, a $128 \times 33{,}278$ Linear layer, is included, and dominates the model's parameter count the way a large softmax head does in any real word-level language

model. Coefficients ($\lambda_w = 0.1$, $\lambda_a = 0.05$, $\lambda_c = 0.1$) were screened at a reduced step budget to avoid collapse, then validated at 3000 training steps (≈ 12 passes over the training corpus), three seeds per regime. Table 5 (main text) reports the result.

Marginal reinforcement replicates the char-level probe's split exactly, and this time with a real significance test. Weight sparsity reinforces: joint additive's 68.94% exceeds weight-only's 66.51% by $+2.43$ points (Welch's t-test, $t = -12.72$, $p = 0.0036$; this and the activation-sparsity test below extend the paper's marginal-sparsity significance family from 12 to 14 tests, Appendix L). Activation sparsity instead reverses: joint additive's 76.63% is *lower* than activation-only's 77.26% alone, a -0.63-point gap ($t = 1.47$, $p = 0.27$, not individually significant, but directionally the same reversal the char-level probe shows on this operand). This is the same qualitative split (weight reinforces, activation reverses) observed in both non-CNN probes so far, now confirmed on a real benchmark rather than a single seed on a toy corpus.

Aggregate effective-compute reinforcement fails again, at nearly the identical magnitude. The naive independence estimate from the two solo regimes is $(100 - 66.51\%) \times (100 - 77.26\%) \approx 7.62\%$; joint additive's actual effective-MAC (8.48%) is 11.4% relatively *worse* – remarkably close to the char-level probe's own 11.6% relative shortfall, a fourth documented exception alongside MobileNetV2/CIFAR-10 and the ReLU-ViT probe, and now the second data point suggesting language modeling specifically, rather than any one corpus or tokenization, resists effective-compute reinforcement.

The coupled objective again beats additive, and this time the result is statistically significant – but it does not repeat the char-level probe's more dramatic claim. Joint coupled's validation loss (5.7235) is significantly lower (better) than joint additive's (5.7510; $t = 6.49$, $p = 0.0038$, uncorrected single comparison, not part of the paper's main significance families). This confirms the char-level probe's central anomaly – the coupled objective does not underperform additive here – with real statistical power for the first time. Two things do *not* replicate, though: joint coupled does not beat the unregularized baseline (val. loss 5.7235 vs. baseline's 5.6046; every regularized regime costs some validation performance here, $t = -25.08$ for coupled vs. baseline, $p < 0.0001$), and joint coupled does not reach the lowest effective-MAC fraction of the five regimes – joint additive does (8.48% vs. coupled's 9.24%). The char-level probe's headline claim was that coupled training beats *every* regime, including a dense baseline; at this larger scale, the more modest and more robust version of the claim survives (coupled beats additive, significantly) while the more dramatic version (coupled beats everything) does not.

We treat this as a validated, not preliminary, result: three seeds, a standard benchmark, and Welch's-test significance for the finding that matters most (coupled beats additive here rather than losing to it substantially, as it does throughout the rest of this paper). The coefficients are still only screened rather than swept, and a single architecture/corpus pair is not the same as a systematic study of language modeling the way this paper's vision settings are – but this is no longer a single anecdote either.

Q A Direct Causal Test of Proposition 2

Every reinforcement result in this paper is correlational in a specific sense: joint training applies both regularizers *simultaneously and continuously*, so a weight and its downstream activation become sparse together, gradually, under ongoing gradient pressure from both penalties at once. This leaves open whether Proposition 2's weight-to-activation direction (a smaller-variance pre-activation is easier for ReLU to zero) is doing real causal work, or whether joint training's marginal-sparsity gains are better explained as simply more total regularization pressure landing on a network already adapting to it. We test this directly with a manipulation the ongoing-joint-training design cannot distinguish on its own: *permanently and structurally* fixing weight sparsity *before* activation regularization ever begins, rather than letting the two develop together.

Design. We train weight-only ($\lambda_w = 0.3$) to the standard 10-epoch protocol, single seed, then extract two boolean masks from the converged weights: a *real* mask (each layer's near-zero weights, the same relative-threshold convention as `weight_sparsity_pct`) and a *shuffled* mask (the identical per-layer *count* of frozen positions, but placed uniformly at random within that layer, so it typically zeros weights of ordinary, not small, magnitude). Starting from the same converged weight-only

Table 45: Causal weight-freeze test, ResNet-18/CIFAR-10, single seed. All three continuation conditions start from the identical weight-only-converged checkpoint (86.73% accuracy, 88.69% weight sparsity, 49.80% activation sparsity) and run 10 further epochs at $\lambda_w = 0$, $\lambda_a = 0.05$.

Condition	Accuracy	Weight Sp.	Activation Sp.
Frozen, real mask	88.80%	89.82%	62.27%
Frozen, shuffled mask	86.02%	92.37%	59.20%
No freeze (control)	$\mathbf{91.12}\%$	43.72%	$\mathbf{66.88}\%$
Activation-only, from scratch	80.29%	9.98%	80.28%

model, we continue training for 10 further epochs under three conditions, $\lambda_w = 0$ and $\lambda_a = 0.05$ throughout: (i) **frozen-real** – the real mask's positions are reset to exactly zero after every optimizer step, permanently; (ii) **frozen-shuffled** – the shuffled mask's positions are frozen the same way; (iii) **no-freeze** – an otherwise identical continuation with no freezing at all, isolating the effect of merely *starting* from a weight-sparse checkpoint. Plain activation-only trained from scratch (10 epochs, same seed) is included for reference.

This particular causal manipulation does not reproduce the reinforcement effect – if anything, it runs backward. No-freeze reaches *both* higher accuracy and higher activation sparsity than either frozen condition (66.88% vs. 62.27% real and 59.20% shuffled): permanently removing weights from optimization, even along a realistic redundancy structure, costs the network more in lost representational flexibility than it gains in activation-sparsity reinforcement, if it gains anything at all. Comparing real to shuffled in isolation gives a small, single-seed effect in the predicted direction (real exceeds shuffled by $+3.07$ activation-sparsity points and $+2.78$ accuracy points) – weak evidence that *which* weights are frozen matters somewhat, consistent with freezing already-small, genuinely redundant weights being less structurally damaging than freezing an equally-sized random subset – but this is dominated by the much larger, opposite-signed cost of freezing at all. None of the three continuation conditions reaches plain activation-only's activation sparsity (80.28%) starting from scratch.

We do not read this as evidence against reinforcement generally, which this paper establishes through many other, correlational routes; we read it as evidence that *this specific* causal manipulation – a permanent, hard structural constraint imposed before the fact – is a poor model of what joint training actually does, which is impose a continuous, soft gradient pressure on both operands *simultaneously*, letting each adapt to the other's ongoing state rather than to a fixed prior one. Proposition 2's mechanism may still operate moment-to-moment during ordinary joint training without surviving translation into a permanent freeze imposed in advance. This is a single-seed exploratory test, not a validated result, and a natural next step would be a softer causal intervention (e.g. a decaying rather than permanent freeze, or freezing only a subset of layers) that more closely approximates joint training's actual dynamics.

R Is the Coupled Objective's Failure Specific to Hoyer?

Section 4.2 diagnoses the coupled objective's underperformance as substantially a convergence-speed artifact of the product rule, but this leaves a further question open: is the failure specific to using Hoyer's concentration ratio, $H(x) = (\sum_i |x_i|)^2/(N\sum_i x_i^2)$, as the per-operand factor, or would any product-form coupling of weight and activation sparsity fail similarly? Hoyer measures concentration (how unevenly mass is distributed across coordinates), not density (what fraction of coordinates are effectively zero) – the two coincide at the extremes but differ in between, and it is the gradient of the coupled *concentration* product, not of density, whose throttling behavior Appendix G measures. We test whether a genuinely density-aligned surrogate changes the picture.

A density surrogate. We replace $H(\cdot)$ with $\hat{\rho}(x) = \mathrm{mean}_i\left[x_i^2/(x_i^2 + \tau^2)\right]$ ($\tau = 0.01$), a smooth relaxation of the fraction-nonzero indicator: $\hat{\rho} \to 1$ for $|x_i| \gg \tau$ and $\hat{\rho} \to 0$ for $|x_i| \ll \tau$, unlike Hoyer's scale-invariant ratio of norms. We use this measure in place of Hoyer for all four regularized regimes (including the coupled product, $\hat{\rho}_W(W_l) \cdot \hat{\rho}_A(A_l)$), screen coefficients at 5 epochs single seed to avoid collapse ($\lambda_w = 0.2$, $\lambda_a = 0.05$, $\lambda_c = 0.2$, the latter chosen to match joint additive's

own effective-MAC fraction), then validate at the standard 10-epoch, 3-seed protocol. Table 3 (main text) reports the result.

The catastrophic gap disappears. Under Hoyer at the identical n=3 protocol, joint coupled costs additive 14.23 points ($t = -9.87$, $p < 0.0001$; Table 1's original 3-seed validation). Under the density surrogate (Table 3, main text), the gap is 0.30 points and the two are statistically indistinguishable (Welch's t-test, $t = 0.15$, $p = 0.89$) – not a smaller version of the same failure, but no detectable failure at all on accuracy. The trade-off shifts location rather than disappearing: density-coupled reaches substantially lower weight sparsity than additive (57.12% vs. 81.65%) and higher activation sparsity (81.31% vs. 72.61%), the same lopsided allocation pattern Hoyer's coupled objective shows throughout this paper, and its effective-MAC fraction is slightly *worse* than additive's (6.50% vs. 5.58%) rather than matching or beating it – so the coupled objective's asymmetric sparsity allocation is not Hoyer-specific, but the severe *accuracy* collapse is.

This directly answers the open question from Section 4.2: the product rule's gradient-throttling mechanism (Appendix G) depends on which per-operand measure is coupled, not only on the fact of coupling itself. Hoyer's concentration ratio and this density surrogate have different gradient landscapes near sparse configurations, and only one of them produces the severe, slowly-recovering accuracy penalty studied throughout this paper. We do not have a full mechanistic account of why – a natural next step is instrumenting this density-coupled run's gradient trajectories the same way Appendix G does for Hoyer – but the result itself is a single, clean, $n = 3$-validated fact: this paper's coupled-objective failure story is a fact about Hoyer specifically, not about product-form coupling of sparsity measures in general.

S Generalization to Deployed Edge-AI Models

Every setting so far combines an activation regularizer with a *differentiable* weight-sparsity mechanism – either per-element Hoyer (Table 1) or the per-channel Hoyer analogue of Appendix D.6. This appendix tests external validity along a different axis: does the same qualitative question – does pushing activation sparsity and weight sparsity together reinforce or compete, relative to either alone? – replicate when the weight-sparsity mechanism is instead *hard, discrete channel pruning*, and the networks are four real production architectures from an industrial edge-AI model zoo rather than standard vision-research benchmarks?

Each of the four studies below independently repeats the same pilot design on a different model-zoo project (arrhythmia ECG classification, PlantVillage leaf-disease classification, Google Speech Commands keyword spotting, and Visual Wake Words person detection), each on its own dedicated branch. Activation sparsity uses the same Hoyer-Square activity regularizer (Yang et al., 2020) attached to every ReLU layer as the main text (in its per-element-normalized form for three of the four projects; Section S.3 notes the exception), at a strength λ independently calibrated per project in each project's own prior single-axis study. Weight sparsity instead uses Network Slimming (Liu et al., 2017): an L_1 penalty on the scale (γ) parameter of each targeted BatchNormalization layer during training, after which the lowest-$|\gamma|$ fraction p (the *prune target*) of that layer's output channels is physically removed and the model rebuilt smaller, followed by a short fine-tuning phase to recover accuracy. This is a fundamentally different weight-sparsification mechanism from anything else in this paper – irreversible and magnitude-ranked rather than differentiable and jointly co-trained with the activation penalty – so agreement or disagreement with the main text's reinforcement finding here is independent evidence, not a replication of the same mechanism at a new operating point. All four studies are float-only (no quantization or target-hardware conversion), so activation-sparsity percentages are internally consistent within a project but not comparable across projects or to any hardware-measured number elsewhere; each project's pruning is scoped to its backbone's depthwise/separable convolution blocks, excluding the stem and the final feature layer that feeds the classifier head (resizing the classifier along with the pruned layer was judged unnecessary implementation risk for these pilots). Every study reports four to eight grid points spanning `baseline`, `activation-only`, `structured-only`, and one or more `joint` configurations at matched or lighter regularizer/prune-target pairs.

Table 46: VWW backbone ($\alpha = 0.25$ MobileNetV1-style) architecture. Each `conv_*/separable_*` row is immediately followed by its own BatchNormalization and clipped ReLU (omitted for brevity; their parameters are included in the running total).

Layer	Type	Output shape	Params (layer + BN)
input / rescaling	Input, $\div 255$	96×96×3	0
conv_0	Conv2D, stride 2	48×48×8	248
conv_1	Conv2D	48×48×16	1,216
conv_2	Conv2D, stride 2	24×24×32	4,736
conv_3	Conv2D	24×24×32	9,344
separable_4	SeparableConv2D, stride 2	12×12×64	2,592
separable_5	SeparableConv2D	12×12×64	4,928
separable_6	SeparableConv2D, stride 2	6×6×128	9,280
separable_7 – separable_11 (5×)	SeparableConv2D	6×6×128	5×18,048
separable_12	SeparableConv2D, stride 2	3×3×256	34,944
separable_13	SeparableConv2D + global-avg-pool	256	68,864
predictions	Dense (logits)	2	514
	Total (trainable / non-trainable)		**226,906 (223,914 / 2,992)**

Table 47: VWW: full results. Accuracy drop is relative to baseline; parameter reduction relative to baseline's 226,906 params.

Tag	λ	p	Accuracy	Acc. drop	Act. sparsity	Params (reduction)
baseline	0	0%	89.14%	–	51.9%	226,906 (–)
activation-only	3.6	0%	87.32%	1.82pt	80.1%	226,906 (0%)
activation-light	1.5	0%	88.17%	0.98pt	71.1%	226,906 (0%)
structured-only	0	40%	86.11%	3.04pt	54.0%	112,879 (50.3%)
structured-light	0	20%	88.45%	0.69pt	52.9%	164,162 (27.7%)
joint	3.6	40%	85.25%	3.90pt	**77.6%**	**112,879 (50.3%)**
joint-medium	2.5	30%	87.06%	2.08pt	74.7%	137,836 (39.3%)
joint-light	1.5	20%	87.92%	**1.22pt**	69.4%	164,162 (27.7%)

S.1 Visual Wake Words (MobileNetV1-style, $\alpha = 0.25$)

VWW is a binary person/no-person image-classification task (96×96 RGB, COCO-derived). The network is a MobileNetV1-style backbone at width multiplier $\alpha = 0.25$, ImageNet-pretrained, with a 2-class head, a strict feed-forward chain with no skip connections (Table 46).

Channel pruning targets `separable_4`–`separable_12` (9 of 10 separable blocks); `separable_13` is excluded because the classifier head reads it directly. Table 47 reports all eight grid points (baseline accuracy 89.14%).

The two axes are complementary. Activation-only alone costs 1.82 points and structured-only alone costs 3.04 points; a naive sum predicts ≈ 4.86 points for both together. The aggressive joint configuration costs only 3.90 points – less than the naive sum – while reaching 77.6% activation sparsity (close to activation-only's own 80.1%) and the full 50.3% parameter cut structured-only achieves alone. Halving both knobs (joint-light) reduces the cost further to 1.22 points while retaining 69.4% activation sparsity and a 27.7% parameter cut; the three joint points trace a smooth, gradually-degrading curve with no sign of a collapse anywhere in the range tested. **joint-light is the recommended operating point.**

S.2 PlantVillage (MobileNetV1-style, $\alpha = 0.5$)

PlantVillage is a 38-class leaf-disease image-classification task (224×224 RGB). The network is the same MobileNetV1-style backbone family at $\alpha = 0.5$ (twice VWW's width), with a deeper head than VWW: the backbone feeds a 512-unit dense block (its own BN + ReLU) and dropout before the 38-way logits layer (Table 48).

The aggressive joint configuration is antagonistic, not complementary. Activation-only costs 1.42 points and structured-only costs 1.40 points alone; a naive sum predicts ≈ 2.81 points. The aggressive joint configuration instead costs 19.58 points – an order of magnitude worse, the opposite of VWW's pattern. Phase-1 training (both regularizers active) is smooth throughout; the collapse occurs precisely

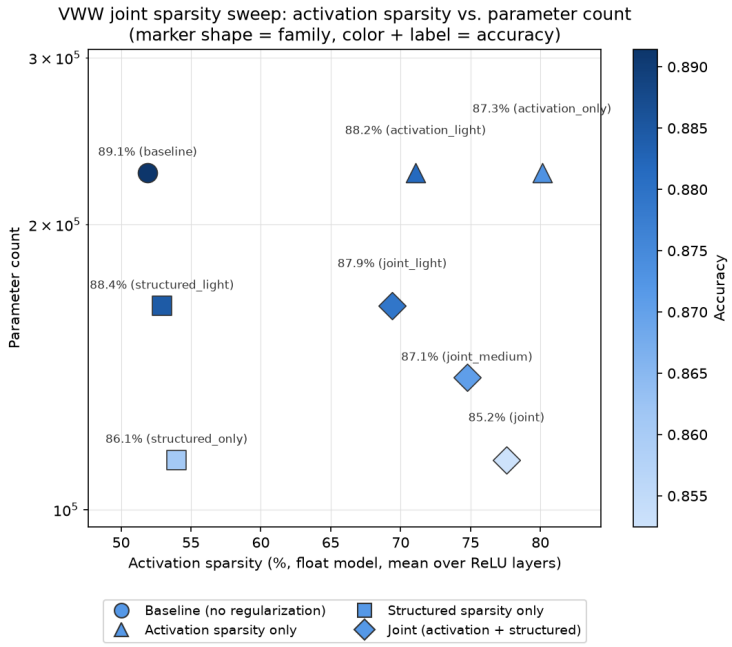


Figure 10: VWW: activation sparsity vs. parameter count across all eight configurations. Marker shape encodes family (circle: baseline, triangle: activation-only, square: structured-only, diamond: joint); color/label encode accuracy.

Table 48: PlantVillage backbone ($\alpha = 0.5$ MobileNetV1-style) architecture; BN+ReLU after each conv/separable row omitted for brevity.

Layer	Type	Output shape	Params (layer + BN)
input / rescaling	Input, $\div 255$	224×224×3	0
conv_0	Conv2D, stride 2	112×112×16	496
conv_1	Conv2D	112×112×32	4,736
conv_2	Conv2D, stride 2	56×56×64	18,688
conv_3	Conv2D	56×56×64	37,120
separable_4	SeparableConv2D, stride 2	28×28×128	9,280
separable_5	SeparableConv2D	28×28×128	18,048
separable_6	SeparableConv2D, stride 2	14×14×256	34,944
separable_7 – separable_11 (5×)	SeparableConv2D	14×14×256	5×68,864
separable_12	SeparableConv2D, stride 2	7×7×512	135,424
separable_13	SeparableConv2D + global-avg-pool	512	268,800
fc_1	Dense + BN + ReLU	512	264,704
predictions	Dense (logits, after dropout)	38	19,494
	Total (trainable / non-trainable)		**1,156,054 (1,149,046 / 7,008)**

at the pruning step, where validation accuracy drops from 98.3% to 59.9% in a single step and the standard 5-epoch fine-tune recovers only to 78.9% – a pruning-recovery failure, not an unstable joint objective. Quadrupling the fine-tune budget (`joint-long-finetune`) recovers most but not all of this (93.48%, a 6.24-point cost, still larger than either mechanism alone), confirming the 19.58-point figure is substantially an under-recovery artifact rather than irreversible channel damage. **Going lighter instead avoids the problem almost entirely**: joint-light (λ=15, p=20%), using the same modest 5-epoch fine-tune that left the aggressive configuration badly under-trained, reaches 97.81% (1.92-point cost) – better on every axis than joint-long-finetune achieved with four times the recovery budget at the aggressive setting. joint-light is the clear recommended operating point; the aggressive corner is a genuine trap for this model, not merely a less-efficient use of the same accuracy budget.

S.3 Speech Commands (DS-CNN)

Speech Commands is a 12-class keyword-spotting task on $49\times10\times1$ MFCC-style features. The network, DS-CNN, is the smallest studied here (22,668 params): a stem convolution, three depthwise-

Table 49: PlantVillage: full results (baseline 99.72%). `joint` uses the default 5-epoch post-prune fine-tune; `joint-long-finetune` is the identical (λ, p) with a 20-epoch fine-tune.

Tag	λ	p	Accuracy	Acc. drop	Act. sparsity	Params (reduction)
baseline	0	0%	99.72%	–	51.7%	1,156,054 (–)
activation-only	36	0%	98.31%	1.42pt	83.1%	1,156,054 (0%)
activation-light	15	0%	98.86%	0.87pt	76.8%	1,156,054 (0%)
structured-only	0	40%	98.32%	1.40pt	56.9%	711,624 (38.4%)
structured-light	0	20%	99.50%	0.22pt	55.0%	913,771 (21.0%)
joint	36	40%	80.15%	**19.58pt**	81.4%	711,624 (38.4%)
joint-long-finetune	36	40%	93.48%	6.24pt	83.2%	711,624 (38.4%)
joint-light	15	20%	97.81%	**1.92pt**	75.9%	913,771 (21.0%)

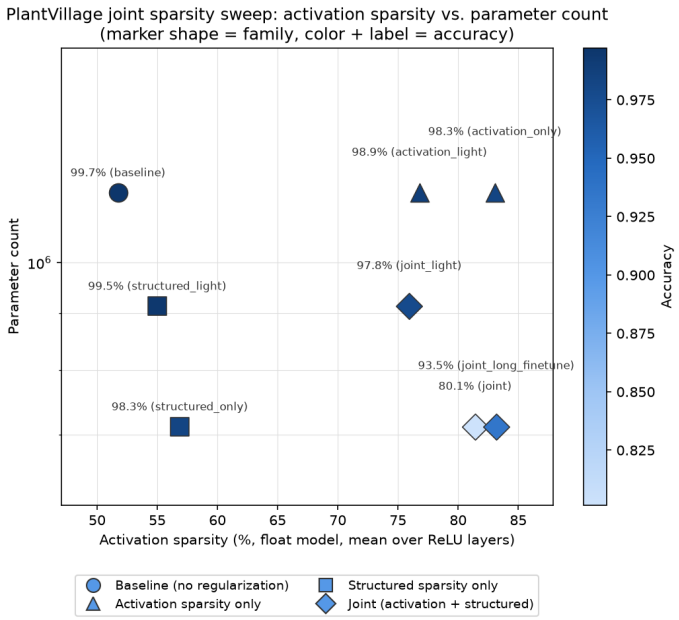


Figure 11: PlantVillage: activation sparsity vs. parameter count. The two diamonds at matched parameter count are the under-trained `joint` (80.1% accuracy, 5 fine-tune epochs) and `joint-long-finetune` (93.5%, 20 epochs).

separable blocks, a fourth ending in global-average-pooling, and a 1×1 pointwise-convolution classifier head (Table 50). Unlike VWW/PlantVillage's explicitly-named pretrained backbone, this model's layers are auto-named by Keras at build time; because the sweep rebuilds the model once per grid point within one Python process, the auto-naming counter shifts between builds and silently breaks the pruning code's layer lookup on the third build onward – fixed by calling `clear_session()` immediately before each build. Also unlike the other three projects, the activation regularizer here is the *raw*, non-normalized Hoyer-Square (the single-axis study found the normalized form's per-tensor-size scaling made it hard to get any effect on this model's uniformly small (64-channel) tensors), so its λ ($\sim 10^{-4}$) is not on a comparable scale to the other three projects' λ.

The sign of the interaction depends on where in (λ, p)-space one looks. At the aggressive setting, activation-only costs 0.89 points and structured-only costs 1.34 points; a naive sum predicts ≈ 2.23 points, while joint costs only 1.84 – complementary, matching VWW's pattern. At the light setting, activation-light costs 0.62 points and structured-light a negligible 0.02 points (naive sum ≈ 0.64); joint-light instead costs 0.92 points, a real 0.28-point *excess* over the naive prediction – the opposite sign from the aggressive corner, and consistent with neither VWW's (complementary throughout)

Table 50: DS-CNN architecture (production config: 64 filters, pointwise-conv head); BN+ReLU after each conv/separable row omitted for brevity.

Layer	Type	Output shape	Params (layer + BN)
input / rescaling	Input, ÷255	49×10×1	0
conv2d	Conv2D, stride 2	25×5×64	1,920
separable_conv2d	SeparableConv2D	25×5×64	4,992
separable_conv2d_1	SeparableConv2D	25×5×64	4,992
separable_conv2d_2	SeparableConv2D	25×5×64	4,992
separable_conv2d_3	SeparableConv2D + global-avg-pool	64	4,992
reshape	Reshape	1×1×64	0
conv2d_1	Conv2D (pointwise, logits)	1×1×12	780
		Total (trainable / non-trainable)	**22,668 (22,028 / 640)**

Table 51: Speech Commands: full results (baseline 96.03%).

Tag	λ	p	Accuracy	Acc. drop	Act. sparsity	Params (reduction)
baseline	0	0%	96.03%	–	64.7%	22,668 (–)
activation-only	10^{-4}	0%	95.14%	0.89pt	80.4%	22,668 (0%)
activation-light	5×10^{-5}	0%	95.41%	0.62pt	78.8%	22,668 (0%)
structured-only	0	40%	94.69%	1.34pt	65.0%	12,944 (42.9%)
structured-light	0	20%	96.01%	0.02pt	63.7%	17,468 (22.9%)
joint	10^{-4}	40%	94.19%	1.84pt	**78.5%**	**12,944 (42.9%)**
joint-light	5×10^{-5}	20%	95.11%	**0.92pt**	77.6%	17,468 (22.9%)

nor PlantVillage's (antagonistic only at the aggressive setting) pattern. The two joint points form a genuine two-point Pareto frontier rather than one dominating the other: joint reaches more activation sparsity and parameter reduction (78.5%/42.9%) than joint-light (77.6%/22.9%) at exactly double the accuracy cost. Neither point resembles PlantVillage's collapse; this model's much smaller per-layer channel counts (64, versus VWW's up to 256 and PlantVillage's up to 512) may leave less room for the two regularizers' pruning-relevant signals to diverge.

S.4 ECG Arrhythmia Classification (custom CNN)

This is a 3-class ECG-beat classification task on 36×32×1 spectrogram-like input. The network is a small custom hardware-targeted CNN (23,363 params, comparable in scale to Speech Commands): a stem convolution, two depthwise-separable blocks with max-pooling, a third separable block whose ReLU is deferred until after global-average-pooling, and a two-layer dense head (Table 52). This project names its BatchNormalization layers by a different convention (`{block}_bn` rather than the other three projects' `{layer}/BN`), so the pruning implementation locates each target's paired BN layer *positionally* (the layer immediately following it) rather than by name suffix, generalizing the Network Slimming implementation to a naming convention it was not originally written for.

Activation regularization improves accuracy here, not just sparsity. Activation-only beats baseline by 1.41 points (93.85% → 95.26%) while gaining 30.5 activation-sparsity points, matching this project's own single-axis finding that mild regularization can act as a generalization aid under its class-weighted loss rather than a pure capacity reduction – behavior none of the other three projects showed. **The joint configuration also beats baseline** (94.43%, a 0.57-point gain, 74.6% activation sparsity, the full 25.7% parameter cut structured-only achieves alone): structured pruning's own small cost (0.41pt) is more than offset by activation regularization's gain. **A fourth distinct interaction pattern: the aggressive configuration strictly dominates the light one.** Unlike VWW (light simply costs less), PlantVillage (light is a strict win only because aggressive is a trap), or Speech Commands (a genuine two-point trade-off), here joint beats joint-light on *every* axis at once – higher accuracy, higher activation sparsity, and a larger parameter cut – so there is no trade-off to make between them. This is consistent with this project's own single-axis study, which found no regularizer strength ever caused an accuracy collapse even at 30–100× the strength that destroyed the other three models; if stronger regularization simply keeps paying off well beyond the range tested here, that would explain why this pilot's aggressive joint corner has not yet found a downside.

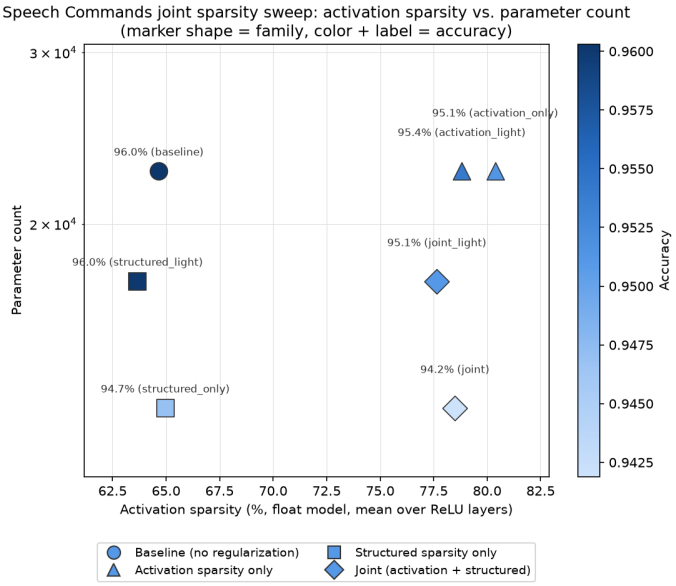


Figure 12: Speech Commands: activation sparsity vs. parameter count across all seven configurations.

Table 52: Arrhythmia-classification architecture. `block3_sepconv` is notably not followed by an activation – its ReLU is deferred until after global-average-pooling.

Layer	Type	Output shape	Params (layer + BN)
input	Input	36×32×1	0
stem_conv	Conv2D	36×32×16	464
block1_sepconv + maxpool	SeparableConv2D, BN	18×16×32	1,040
block2_sepconv + maxpool	SeparableConv2D, BN	9×8×64	3,104
block3_sepconv + GAP + ReLU	SeparableConv2D (no act.), BN	128	10,304
dense	Dense + ReLU	64	8,256
dense_1	Dense (logits)	3	195
	Total (trainable / non-trainable)		**23,363 (22,883 / 480)**

S.5 Summary: A Fourth Reinforcement Pattern, and No Single Rule

Across four independent, non-synthetic production architectures spanning 22.7K to 1.16M parameters, two entirely different mechanisms for inducing weight sparsity (this appendix's hard, discrete Network Slimming vs. the main text's differentiable Hoyer penalty), and two entirely different measures of what "joint" means (a single aggressive grid point vs. this paper's coefficient-swept regimes), the same qualitative conclusion holds: **the interaction between activation sparsity and weight sparsity is conditional on the specific model and operating point, not a fixed property of jointly regularizing the two.** VWW is complementary throughout the range tested; PlantVillage is antagonistic by an order of magnitude at aggressive settings (a pruning-recovery failure, not training instability) yet resolves almost entirely at lighter settings; Speech Commands is complementary at aggressive settings but mildly antagonistic at light settings, a genuine sign flip within a single model; and ECG arrhythmia classification is uniquely favorable, with joint sparsification *improving* accuracy over baseline and the aggressive configuration strictly dominating the lighter one. No two of the four projects show the same pattern, reinforcing this paper's central claim – that reinforcement between weight and activation sparsity is real but conditional – with evidence from a completely independent experimental lineage: real deployed hardware-target networks, a discrete rather than differentiable weight-sparsity mechanism, and pilots run without knowledge of this paper's own generalization results.

Table 53: Arrhythmia classification: full results (baseline 93.85%). A *negative* accuracy drop means the configuration beat baseline.

Tag	λ	p	Accuracy	Acc. drop	Act. sparsity	Params (reduction)
baseline	0	0%	93.85%	–	48.7%	23,363 (–)
activation-only	3	0%	95.26%	−1.41pt	79.2%	23,363 (0%)
activation-light	1.5	0%	93.73%	0.12pt	73.0%	23,363 (0%)
structured-only	0	40%	93.44%	0.41pt	45.5%	17,370 (25.7%)
structured-light	0	20%	93.35%	0.50pt	47.3%	20,330 (13.0%)
joint	3	40%	**94.43%**	**−0.57pt**	**74.6%**	**17,370 (25.7%)**
joint-light	1.5	20%	94.01%	−0.15pt	70.8%	20,330 (13.0%)

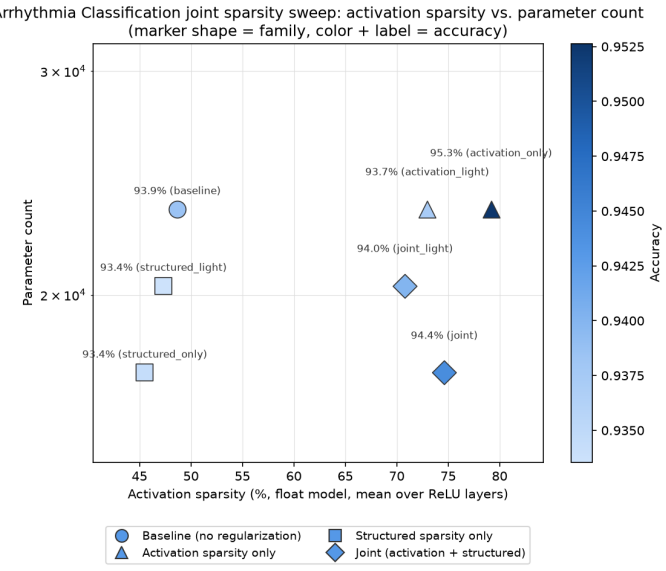


Figure 13: Arrhythmia classification: activation sparsity vs. parameter count across all seven configurations.