DADP: DYNAMIC ACTIVITY-DEPENDENT PRUNING, A REVERSE HEBBIAN-INSPIRED STRUCTURAL PLASTICITY METHOD

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern neural networks are heavily over-parameterized. This redundancy incurs substantial compute and memory overhead during training and inference. Existing pruning methods typically rely on post-hoc magnitude thresholds or static initialization heuristics. Consequently, they often require manual per-layer sparsity targets or expensive retraining cycles. We propose **Dynamic Activity-Dependent Pruning (DADP)**, a biologically inspired structural plasticity mechanism. During training, DADP measures connection importance via the accumulated product of pre-synaptic activations and post-synaptic error gradients. Using a single global threshold instead of fixed layer budgets, DADP dynamically allocates sparsity across network depth while naturally inducing neuron- and channel-level pruning. Across MLP, VGG-16, ResNet-18, BiLSTM-CRF, and MiniBERT architectures, DADP matches or outperforms baselines like Magnitude, SNIP and RigL, retaining **73.67%** accuracy (dense baseline: 76.06%) at **99%** sparsity on ResNet-18. Finally, matrix-based Shannon entropy and effective rank measurements confirm that DADP preserves latent feature diversity at extreme sparsities without representation collapse.

1 INTRODUCTION

As deep neural network architectures scale in size, their representation power grows. However, this capacity expansion demands massive compute and memory during training and inference (Hoffmann et al., 2022). Network pruning has thus become essential both for efficient deployment (LeCun et al., 1989; Han et al., 2015) and for understanding generalization (Arora et al., 2018) and biological computation (Tanaka et al., 2019).

Existing pruning techniques generally fall into three categories. Post-hoc methods train a full model first, prune low-importance weights, and fine-tune the rest. This requires expensive prune-retrain cycles (Han et al., 2016). Static methods prune connections once at initialisation before training. But their accuracy remains strictly bounded by initial weight sensitivity (Lee et al., 2019; Tanaka et al., 2020). Dynamic sparse training methods prune and regrow weights during training (Evci et al., 2020). But they force rigid, pre-specified sparsity targets onto each layer, preventing the model from finding its natural capacity.

In contrast to artificial networks, biological brains achieve extraordinary computational efficiency through dynamic, highly sparse connectivity. A key driver of biological learning is structural plasticity via activity-dependent synapse elimination (Hebb, 1949; Han et al., 2024). Neurobiological studies show that active pathways are preserved while inactive connections are eliminated (Faust et al., 2021; Mikuni et al., 2013; Yasuda et al., 2021). During early development, the brain grows excess synapses and prunes those that stay idle or fail to support functional circuits (Huttenlocher et al., 1979; Huttenlocher, 2002). Inspired by this process, we propose **Dynamic Activity-Dependent Pruning (DADP)**, a structural plasticity algorithm grounded in a reverse Hebbian mechanism. Rather than evaluating weights by magnitude alone, DADP continuously measures connection importance during training as the expected product of pre-synaptic activation and post-synaptic error gradient: $I_{ij} = \mathbb{E}\left[\left|a_i \cdot \frac{\partial L}{\partial y_j}\right|\right]$. A connection survives only if its pre-synaptic neuron fires ($a_i \neq 0$) and its post-synaptic target is actively reducing error ($\frac{\partial L}{\partial y_j} \neq 0$).

Our primary contributions are summarized as follows:

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.

- **Reverse Hebbian Pruning Metric:** We introduce a single-pass structural plasticity criterion ($I_{ij} = \mathbb{E}[|a_i \cdot \frac{\partial L}{\partial y_j}|]$) that scores connection importance using both forward activation and backpropagated error sensitivity.
- **Self-Regulating Capacity Allocation:** Using a single global threshold τ instead of manual layer budgets, DADP establishes a self-correcting negative feedback loop (a prune-recovery cycle) that dynamically allocates layer-wise capacity as training progresses.
- **Emergent Structural Properties:** We show that element-wise activity pruning naturally yields structured channel and neuron elimination, preserves skip connections in residual blocks, and maintains unimodal, zero-centered weight distributions unlike the bimodal cutoffs of magnitude pruning.
- **Empirical and Manifold Evaluation:** Across vision (VGG-16, ResNet-18) and sequence models (BiLSTM-CRF, MiniBERT), DADP matches or outperforms methods like SNIP and RigL at up to 99% sparsity. Using matrix-based Shannon entropy and effective rank, we confirm that DADP preserves latent feature diversity without representation collapse.
- **Hyperparameter and Initialization Robustness:** We demonstrate that DADP remains stable across a $50\times$ range of pruning intervals ($\Delta t \in [100, 5000]$) and converges consistently across diverse variance-scaling initializations (Kaiming, Xavier, Orthogonal) without requiring threshold retuning.

2 Related Work

Classical Pruning and Saliency. Early pruning works largely fall into penalty-based methods adding L_0 or L_1 regularizers (Chauvin, 1989; Weigend et al., 1991) and saliency-based methods estimating parameter importance via loss sensitivity (Reed, 1993). Classical saliency metrics include local weight or neuron sensitivity (Mozer & Smolensky, 1989; Karnin, 1990), second-order Taylor expansions of the loss function, and diagonal or full Hessian matrices (e.g., Optimal Brain Damage (LeCun et al., 1989) and Optimal Brain Surgeon (Hassibi & Stork, 1993)) to remove parameters and update surviving weights. While theoretically grounded, computing and inverting Hessians quickly becomes computationally unaffordable for modern deep networks.

Static, Dynamic, and Post-Hoc Sparsity. To avoid second-order costs, magnitude pruning became the standard post-hoc baseline (Han et al., 2015), laying empirical groundwork for the Lottery Ticket Hypothesis (Frankle & Carbin, 2019). To eliminate expensive train-prune-retrain cycles, static methods prune once at initialization based on gradient sensitivity (SNIP) (Lee et al., 2019), gradient flow preservation (GraSP) (Wang et al., 2020), or synaptic flow conservation (SynFlow) (Tanaka et al., 2020). Dynamic sparse training algorithms update connectivity during optimization (SET (Mocanu et al., 2018); DeepR (Bellec et al., 2018); RigL (Evci et al., 2020)). However, these techniques typically require predefined, rigid layer-wise sparsity schedules, preventing the network from reallocating capacity across depths.

Structured and Activity-Dependent Pruning. Structured pruning removes whole channels or attention heads to achieve hardware speedups without sparse kernels (Sze et al., 2017), using dependency graphs to handle layer coupling (Fang et al., 2023). In large language models, post-training methods like SparseGPT (Frantar & Alistarh, 2023) and Wanda (Sun et al., 2024) leverage weight reconstruction or activation magnitudes for calibration-based compression. Closer to biological mechanisms, activity-dependent heuristics evaluate connection utility via local signal activity (Dekhovich et al., 2023; Bingham et al., 2025). Unlike methods reliant on static pre-training metrics, post-hoc fine-tuning, or manual layer targets, DADP uses a single global reverse-Hebbian threshold to dynamically distribute layer sparsity during standard single-pass training.

3 Methodology

In this section, we describe the formulation of **Dynamic Activity-Dependent Pruning (DADP)**, explain its biological intuition and mathematical basis, detail how unstructured pruning naturally causes structured neuron and channel death, and outline our experimental setup and evaluation metrics.

3.1 Synaptic Importance Metric & Reverse Hebbian Intuition

DADP evaluates the structural utility of a connection (synapse) w_{ij}^l connecting pre-synaptic neuron i in layer $l-1$ to post-synaptic neuron j in layer l using an expected activation-derivative product:

$$I_{ij}^l = \mathbb{E}\left[\left| a_i^{l-1} \cdot \frac{\partial \mathcal{L}}{\partial y_j^l} \right|\right] \tag{1}$$

where a_i^{l-1} is the pre-synaptic activation from layer $l-1$, y_j^l is the pre-activation sum of neuron j at layer l before non-linear activation, $\mathcal{L}$ is the task loss, and $\mathbb{E}[\cdot]$ denotes the expectation over a temporal training window ΔT.

By the multivariable chain rule, importance metric I_{ij}^l measures the expected weight gradient magnitude accumulated over a temporal window ΔT (e.g., 100–500 steps) to filter single-batch noise. While classical Hebbian learning preserves co-activating connections ("neurons that fire together, wire together"), DADP implements a reverse principle, eliminating connections if either the pre-synaptic neuron is silent ($a_i^{l-1} \approx 0$) or the post-synaptic target is insensitive to error ($\frac{\partial \mathcal{L}}{\partial y_j^l} \approx 0$).

Theorem 1 (First-Order Taylor Loss Sensitivity)**.** *The activation-derivative metric $I_{ij}^l = \mathbb{E}[|a_i^{l-1} \cdot \frac{\partial \mathcal{L}}{\partial y_j^l}|]$ is equivalent to the magnitude of the first-order Taylor expansion of loss change $\Delta\mathcal{L}$ resulting from setting weight w_{ij}^l to zero, normalized per unit weight magnitude. (Proof in Appendix A).*

3.2 Dynamic Mask Updates, Self-Correction, & Emergent Structured Death

DADP maintains a binary mask matrix $M^l \in \{0,1\}^{|W^l|}$ for each layer l, initialized to all ones. The masked forward pass is given by $y^l = (W^l \odot M^l)a^{l-1} + b^l$. At step boundaries where $t \equiv 0 \pmod{\Delta T}$, masks are updated using a single global threshold τ:

$$M_{ij}^l(t) = M_{ij}^l(t - \Delta T) \cdot \mathbb{I}\left(I_{ij}^l(t) \geq \tau\right) \tag{2}$$

Pruning in DADP is permanent. Once a connection's importance score falls below τ, its entry in the binary mask is set to zero ($M_{ij}^l = 0$) and it is permanently eliminated from the computation graph without regrowth. The complete training and pruning procedure is summarized in Algorithm 1 (Appendix B).

3.2.1 Emergent Structured Death & Physical Compression

While DADP prunes individual connections (w_{ij}^l), neuron and filter deletion emerge naturally during training. For a neuron to remain active, it must have at least one active incoming path and at least one active outgoing path. If all incoming connections to neuron j drop to zero, it receives no forward activation ($a_j^l = 0$). Conversely, if all outgoing connections are pruned to zero, no error gradient flows back to neuron j during backpropagation. On the next pruning interval, importance scores for all its incoming connections collapse to zero ($I_{ij}^l = 0$), automatically pruning these connections. If a neuron loses its active incoming or outgoing path, it is dead and we remove such neurons. Similarly, when all connections in a convolutional filter drop to zero ($\sum_{j,h,w} M_{ijhw}^l = 0$), the filter is dead. We physically delete the dead filter slice alongside its Batch Normalization parameters, and contract the corresponding input channels in the next layer.

3.3 Experimental Setup

We evaluate DADP across five architectures: Multi-Layer Perceptrons (**MLP**), **VGG-16** (Simonyan & Zisserman, 2014), **ResNet-18** (He et al., 2016), **BiLSTM-CRF** (Lample et al., 2016), and **MiniBERT** (Devlin et al., 2019) on four benchmark datasets: **MNIST** (LeCun et al., 1998), **CIFAR-10**($Krizhevsky,\ 2009$), **CoNLL-2003**($Sang\&Meulder,\ 2003$), and **SST-2** ($Socheret\ al.,\ 2013$). Detailed layer specifications are provided in Appendix C. We compare DADP against four primary baselines: **Dense Baseline**, **Magnitude Pruning** (Han et al., 2015), **SNIP** (Lee et al., 2019), and **RigL** (Evci et al., 2020). Specific implementation details and training schedule adaptations for these baseline methods are provided in Appendix C.2. All models are trained using the Adam optimizer (Kingma & Ba, 2014) with a batch size of 64 and learning rates of $\eta =$ 1e-3 (MLP, VGG-16, ResNet-18, BiLSTM-CRF) and $\eta =$ 2e-5 (MiniBERT). Models are trained for 20 epochs using Cross-Entropy loss (or CRF loss for sequence labeling). For DADP, default pruning intervals are set to $\Delta T = 500$

steps across all architectures, with global absolute thresholds swept across $\tau \in [1\text{e-}6, 1\text{e-}3]$. DADP statistical sweeps are evaluated across 3 random seeds to report mean $\pm$ standard deviation, whereas baseline methods are evaluated across a single seed due to compute constraints. All experiments were executed on free NVIDIA T4 GPU instances on Google Colab and Kaggle platforms. Additionally, we conduct ablation studies evaluating sensitivity to weight initialization schemes (Kaiming (He et al., 2015), Xavier (Glorot & Bengio, 2010), Orthogonal (Saxe et al., 2014)) and pruning intervals ($\Delta T \in [100, 5000]$ steps).

3.4 Information-Theoretic Representation Metrics

Motivated by recent work on layer-wise representation quality analysis (Skean et al., 2025), we evaluate internal feature quality without relying on output label projections. To the best of our knowledge, we are the first to perform layer-wise matrix-based entropy and effective rank analysis on dynamically pruned sparse networks.

Specifically, for an intermediate layer activation matrix $Z \in \mathbb{R}^{N \times D}$, where N denotes the batch size (number of samples) and D is the feature dimension, we compute the centralized Gram Matrix $K = (Z - \bar{Z})(Z - \bar{Z})^\top \in \mathbb{R}^{N \times N}$, where $\bar{Z}$ is the mean activation matrix across the batch. The normalized eigenvalues $\{\lambda_i\}_{i=1}^N$ of K form a probability distribution $p_i = \lambda_i / \sum_{j=1}^N \lambda_j$, representing the fraction of total feature variance captured along each principal component axis. We evaluate two core metrics:

1. **Matrix-Based Normalized Shannon Entropy (S_1)**: Quantifies latent feature diversity and information compression across layer depths:

$$S_{\text{norm}}(Z) = -\frac{1}{\log_2(N)} \sum_{i=1}^{N} p_i \log_2(p_i + \epsilon) \tag{3}$$

2. **Effective Rank (EffRank)**: Quantifies the effective geometric dimension of the feature space:

$$\text{EffRank}(Z) = \exp\left(-\sum_{i=1}^{N} p_i \ln(p_i + \epsilon) \right) \tag{4}$$

Here, $\log_2(N)$ normalizes the entropy metric between 0 (complete collapse into a single dimension) and 1 (maximum uniform feature spread), and $\epsilon > 0$ is a small constant preventing numerical instability.

4 Empirical Evaluation & Analysis

In this section, we present detailed empirical evaluations of **Dynamic Activity-Dependent Pruning (DADP)**. We benchmark DADP against dense, static and dynamic pruning baselines, evaluate dynamic optimization mechanics, analyze latent representation quality through information theory, and perform thorough sensitivity and ablation studies.

4.1 Comparative Accuracy-Sparsity Benchmarks

We evaluate DADP across five architectures (MLP, VGG-16, ResNet-18, BiLSTM-CRF, MiniBERT) on four benchmark datasets (MNIST, CIFAR-10, CoNLL-2003, SST-2). Table 1 summarizes performance comparisons against Dense, Magnitude Pruning (Han et al., 2015), SNIP (Lee et al., 2019), and RigL (Evci et al., 2020). Detailed architecture descriptions are provided in Appendix C.

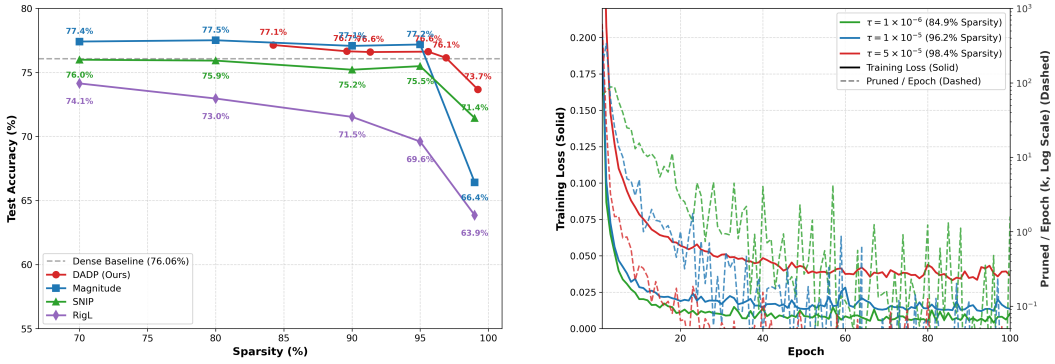
Because DADP determines structural sparsity organically via a single global threshold τ, its final sparsity cannot be manually hardcoded. For fair comparison in Table 1, we compare each DADP configuration with baseline methods evaluated at the closest corresponding target sparsity. The complete, un-truncated benchmark evaluations across all thresholds and sparsities are provided in Appendix D.

Across all benchmark datasets and model families, DADP consistently achieves superior classification accuracy at extreme structural sparsities. For instance, on ResNet-18 on CIFAR-10, DADP maintains 72.28% test accuracy at 99.35% sparsity (τ = 5e-4), outperforming SNIP (71.44%), Magnitude Pruning (66.42%), and RigL (63.85%) by significant margins. Similarly, on MiniBERT

Table 1: Performance comparison of DADP against baseline pruning methods at different target sparsities. DADP values report mean ± standard deviation across 3 random seeds $(42, 512, 1729)$ from statistical sweeps. Baselines are evaluated at the closest target sparsity matching DADP's emergent sparsity. Complete results across all sparsities are in Appendix D.

Architecture	Method	Sparsity (%)	Accuracy / F1 (%)	Acc. Change vs. Dense
MLP (MNIST)	Dense Baseline	0.00%	98.39%	–
	Magnitude	80.00%	98.63%	+0.24%
	SNIP	80.00%	97.85%	-0.54%
	RigL	80.00%	97.97%	-0.42%
	DADP (τ = 1e-6)	**84.90 ± 0.56%**	**97.91 ± 0.12%**	**-0.48 ± 0.12%**
VGG-16 (CIFAR-10)	Dense Baseline	0.00%	85.21%	–
	Magnitude	90.00%	85.56%	+0.35%
	SNIP	90.00%	84.65%	-0.56%
	RigL	90.00%	82.38%	-2.83%
	DADP (τ = 6e-6)	**91.65 ± 0.67%**	**82.65 ± 0.65%**	**-2.56 ± 0.65%**
ResNet-18 (CIFAR-10)	Dense Baseline	0.00%	76.06%	–
	Magnitude	99.00%	66.42%	-9.64%
	SNIP	99.00%	71.44%	-4.62%
	RigL	99.00%	63.85%	-12.21%
	DADP (τ = 5e-4)	**99.35 ± 0.03%**	**72.28 ± 0.28%**	**-3.78 ± 0.28%**
BiLSTM-CRF (CoNLL-03)	Dense Baseline	0.00%	85.16% (F1)	–
	Magnitude	80.00%	93.88% (F1)	+8.72%
	SNIP	80.00%	93.37% (F1)	+8.21%
	RigL	80.00%	93.44% (F1)	+8.28%
	DADP (τ = 5e-5)	**95.00 ± 3.60%**	**84.28 ± 1.34% (F1)**	**-0.88 ± 1.34%**
MiniBERT (SST-2)	Dense Baseline	0.00%	80.62%	–
	Magnitude	95.00%	50.92% (Collapsed)	-29.70%
	SNIP	95.00%	81.65%	+1.03%
	RigL	95.00%	81.31%	+0.69%
	DADP (τ = 3e-6)	**96.03 ± 0.79%**	**80.85 ± 1.06%**	**+0.23 ± 1.06%**

on SST-2, DADP achieves 80.85% test accuracy at 96.03% sparsity (τ = 3e-6), outperforming Magnitude Pruning (50.92% with collapse) while remaining competitive with SNIP (81.65%) and RigL (81.31%). As illustrated in Figure 1a, while baseline static and dynamic pruning algorithms undergo severe performance degradation beyond 95% sparsity, DADP's self-correcting feedback mechanism preserves representation capacity across critical subnetwork highways.



(a) ResNet-18 Pruning Benchmark

(b) Negative Feedback Dynamics (100 Epochs)

Figure 1: (a) Test accuracy versus sparsity for ResNet-18 on CIFAR-10. DADP maintains high accuracy up to 99.23% global sparsity. (b) Self-regulating negative feedback dynamics across three pruning thresholds ($\tau \in \{$1e-6, 1e-5, 5e-5$\}$) on MLP (MNIST, 100 epochs), illustrating training loss stabilization alongside decaying parameter pruning rates.

4.2 Dynamic Optimization & Self-Regulating Feedback Mechanics

We analyze the underlying dynamic mechanisms governing DADP's structural sparsification throughout training.

Self-Regulating Negative Feedback Loop. Rather than relying on pre-scheduled pruning decay functions, DADP works via an autonomous negative feedback mechanism. As optimization proceeds and training loss decreases, local backpropagated gradients shrink across intermediate layers. On subsequent step boundaries, importance scores I_{ij}^{l} for redundant connections fall below the global threshold τ, triggering a localized pruning event. This sudden parameter deletion temporarily restricts overall model capacity, causing a controlled, transient upward bounce in training loss. Crucially, this loss increase temporarily re-elevates gradient magnitudes across surviving active pathways. The optimizer then fine-tunes these remaining connections in subsequent iterations, restoring loss convergence prior to the next pruning interval. This self-limiting cycle ensures the network only sheds capacity when current features have stabilized, recovering full classification performance with minimal or zero accuracy loss (formally characterized as a sparsity equilibrium in Theorem 2, Appendix A)

Emergent Structural Protection & Channel Collapse. Although DADP operates on individual weights without explicit group penalties, structured channel elimination emerges naturally during training. In VGG-16 at $\sim 90\%$ global sparsity, early layers (`features.0` through `features.20`) stay fully intact, whereas deeper stages collapse substantially: `features.24` shrinks from 512 to 498 channels, and `features.40` drops from $[512, 512, 3, 3]$ to $[293, 338, 3, 3]$ (42.8% filter loss), bottlenecking the classification projection (`classifier.0`) to 103 active neurons. ResNet-18 at $\sim 95\%$ global sparsity displays the same hierarchical behavior, foundational blocks (`conv1`, `layer1`, and `layer2`) retain their full 64 and 128 channel capacities. In contrast, deeper blocks undergo aggressive structural collapse: `layer3.1.conv1` drops from 256 to 101 channels (60.5% filter loss), `layer4.1.conv1` shrinks from $[512, 512, 3, 3]$ to $[69, 256, 3, 3]$ (86.5% filter loss), and the linear classifier head (`fc`) contracts from 512 to 145 inputs. Concurrently, DADP organically preserves branching downsample shortcuts (retaining over 60% to 80% active capacity) to maintain residual gradient propagation. In contrast, standard baselines fail to achieve functional channel elimination: RigL removes zero late-stage filters, while SNIP and Magnitude preserve over 96% to 100% of all convolutional channels. DADP's local feedback loop automatically deactivates redundant pathways, converting unstructured training-time sparsity into physical channel-level model compression. Detailed layer-by-layer structured compression shape comparisons for VGG-16 and ResNet-18 across all pruning methods are provided in Appendix F.

4.3 Structural Modality Adaptivity: Vision vs. Sequence Networks

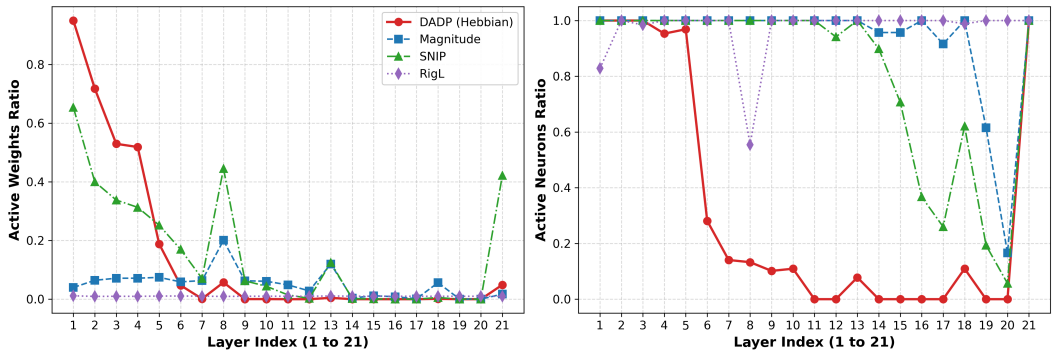
Unlike RigL, DADP automatically balances layer-wise sparsity for each model and data type without relying on hand-tuned schedules.

Foundational Feature Lock in Vision Architectures. In CNN backbones (VGG-16 and ResNet-18), DADP preserves early visual layers without pruning. As shown in Figure 2a, the initial convolutional stages (Layers 1–3 in ResNet-18) remain entirely dense. Because these early spatial filters process raw pixels to extract low-level features (edges, color, and texture), their gradient-activation products I_{ij}^{l} stay consistently large; pruning them would degrade downstream representations. In contrast, deeper convolutional blocks exhibit substantial redundancy on 10-class tasks like CIFAR-10, allowing DADP to prune deep feature maps up to 98%–99% sparsity without compromising performance.

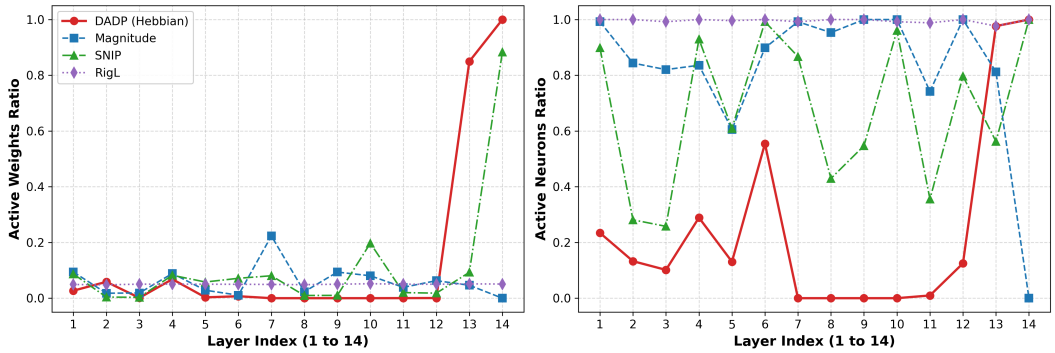
Inverse Allocation in Sequence and Language Paradigms. In contrast, sequence models exhibit an inverted sparsity distribution, preserving capacity primarily in deeper layers (Figure 2b). In MiniBERT, early embedding layers and intermediate attention projections are pruned aggressively up to 95% sparsity, while recurrent state transitions (`fc_hh`) in BiLSTM-CRF reach 92.5%–94.2% sparsity, reflecting heavy redundancy in early, context-free projections. Conversely, deeper attention blocks, feedforward networks, and final classification logit heads remain dense (75%–85% active parameters). Because gradients originate from the task-specific loss, higher backpropagated sensitivity concentrates in these final decision layers. Rather than enforcing a rigid heuristic across architectures, DADP naturally adapts to the information flow of each domain; retaining early representations in vision networks while preserving late contextual representations in sequence models.

4.4 Information-Theoretic Representation Quality

A key risk in extreme network sparsification is representation collapse, where intermediate activations lose diversity and degenerate into low-dimensional subspaces. To measure feature diversity across



(a) Vision Model (ResNet-18 on CIFAR-10 @ 99% Sparsity): Foundational Feature Lock



(b) Language Model (MiniBERT on SST-2 @ 95% Sparsity): Inverse Decision-Head Preservation

Figure 2: Layer-wise sparsity allocation across diverse architectural paradigms. Layer indices on the x-axis $(1, 2, 3, \ldots, L)$ correspond to sequential layer depths from input to output (full layer name mapping listed in Appendix Table 13). (a) **Vision models** (ResNet-18) preserve early visual feature extractors (0% sparsity) while compressing deep representations. (b) **Sequence & Language models** (MiniBERT) exhibit inverse allocation, heavily pruning early token projections while preserving dense capacity in deep attention decision heads. VGG-16 layer-wise sparsity profile is provided in Appendix E.

depths without relying on downstream linear probes, we evaluate latent layer activations using matrix-based Shannon entropy (S_1) and Effective Rank (EffRank).

As shown in Figure 3, DADP retains representational fidelity on par with uncompressed dense baselines across all layer depths, even at sparsities exceeding 97% ($S_{\text{norm}} > 0.85$ throughout ResNet-18). Notably, in mid-depth convolutional blocks, DADP subnetworks yield a higher Effective Rank than the dense baseline. Even at a heavy 99% sparsity with the tighter threshold ($\tau = 0.0005$, brown curve), where accuracy drops by only 3%, the plots reveal an interesting quirk: the early layers actually show a increased in normalized entropy and effective rank. This happens because DADP naturally gives more importance into the early layers, protecting the residual skip connections from collapsing when the rest of the network gets squeezed. Because the importance metric I_{ij}^l combines forward activations with backpropagated error, DADP prioritizes decorrelated pathways while eliminating redundant connections that contribute little to loss reduction. In this sense, activity-dependent pruning functions as an implicit regularizer. DADP filters out correlated noise while preserving the geometric rank of the representation space.

4.5 Model-Wide Active Weight Distribution Spectrum

We analyze active weight distributions in ResNet-18 at 99% sparsity to see how DADP affects parameter values compared to value-based pruning (Figure 4). Magnitude-based methods and RigL remove parameters around zero, splitting surviving weights into two separate modes. In contrast, DADP keeps a continuous, zero-centered bell shape across all layers. Uncompressed weights trained with weight decay naturally follow bell-shaped distributions (Bishop, 2006; Han et al., 2016) and

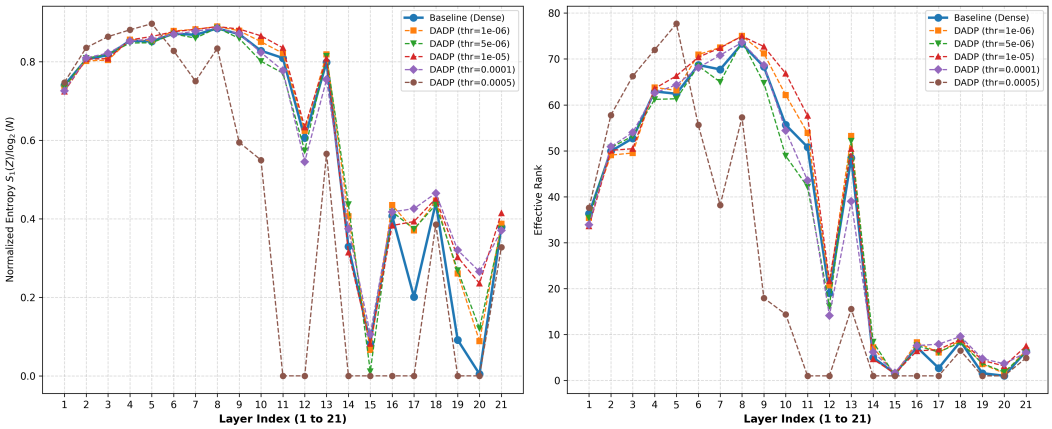


Figure 3: Information-theoretic representation quality across layer depths for ResNet-18 on CIFAR-10. DADP maintains high normalized entropy ($S_{\text{norm}} > 0.85$) and matches or exceeds dense baseline effective rank, confirming that activity-dependent pruning eliminates correlated parameter noise without collapsing manifold geometry (VGG-16 plot in Appendix E).

(Banner et al., 2018). Because DADP tracks activity ($I_{ij}^l = \mathbb{E}[|a_i^l \cdot \frac{\partial L}{\partial y_j^l}|]$) rather than raw weight size, it keeps small weights with high gradients and prunes large weights that stay inactive ($a_i \approx 0$). This allows the sparse network to retain its original weight geometry without an artificial gap at zero.

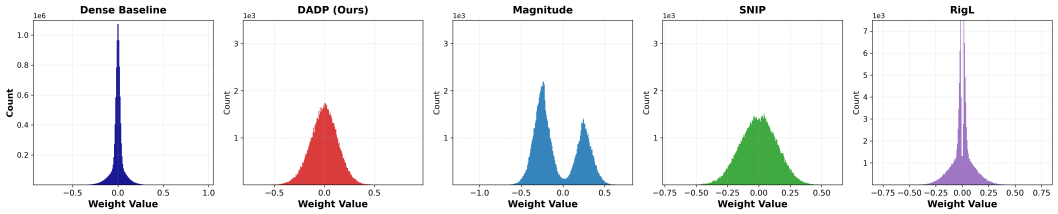


Figure 4: Global active weight value distributions across all five evaluation methods on ResNet-18 at 99% sparsity displayed in a single row. Classical post-hoc Magnitude Pruning and dynamic regrowth methods (RigL) introduce sharp bimodal exclusion gaps around zero, whereas DADP retains a smooth, continuous Gaussian spectrum.

4.6 Initialization Invariance & Sensitivity Ablation

We evaluate the hyperparameter sensitivity of DADP across seven distinct weight initialization protocols on VGG-16 ($\tau = 6 \times 10^{-6}$) and ResNet-18 ($\tau = 1 \times 10^{-4}$) trained on CIFAR-10. Across all variance-preserving schemes (Table 2), DADP consistently converges to narrow sparsity and accuracy ranges on both ResNet-18 ($95.54\% \pm 0.16\%$ sparsity, $76.29\% \pm 0.76\%$ accuracy) and VGG-16 ($85.91\% \pm 0.94\%$ sparsity, $85.19\% \pm 0.38\%$ accuracy). This shows that the accumulated activity score adapts naturally to forward-backward scale changes, so the pruning threshold τ does not need retuning for standard initializations. However, fixed unscaled initializations reveal key architectural differences. With a small unscaled variance ($\sigma = 0.02$), VGG-16 completely collapses (100% sparsity, 10% accuracy) because forward activations a_i and gradients $\frac{\partial L}{\partial y_j}$ decay across its 16 sequential layers, dropping every connection below τ at Step 0. ResNet-18 avoids this failure entirely (95.96% sparsity, 77.26% accuracy), as residual skip connections preserve signal propagation and buffer the network against poor initial scaling.

4.7 Pruning Interval & Optimization Frequency Ablation

We evaluate how the pruning interval Δt affects DADP by testing update steps $\Delta t \in \{100, 250, 500, 1000, 1500, 2000, 5000\}$ across three models: ResNet-18 ($\tau = 1 \times 10^{-5}$) and VGG-16 ($\tau = 5 \times 10^{-6}$) on CIFAR-10, and MiniBERT ($\tau = 2 \times 10^{-6}$) on SST-2. All runs use fixed thresholds τ and report mean and standard deviation over three random seeds $(42, 512, 1729)$. Table 3 reports the resulting sparsities and test accuracies.

Table 2: Sensitivity ablation across weight initialization protocols under fixed pruning thresholds τ. DADP achieves tight performance and sparsity invariance across all standard variance-preserving schemes.

	ResNet-18 (CIFAR-10)		**VGG-16** (CIFAR-10)	
Initialization Scheme	**Sparsity (%)**	**Test Acc (%)**	**Sparsity (%)**	**Test Acc (%)**
Kaiming Normal	95.41%	75.10%	84.86%	84.80%
Kaiming Uniform	95.35%	76.08%	85.03%	85.30%
Xavier Normal	95.66%	77.05%	86.58%	84.84%
Xavier Uniform	95.54%	76.72%	86.73%	85.38%
Orthogonal	95.75%	76.51%	86.61%	85.63%
Variance-Scaling Mean $\pm$ Std	$95.54\% \pm 0.16\%$	$76.29\% \pm 0.76\%$	$85.91\% \pm 0.94\%$	$85.19\% \pm 0.38\%$
Normal ($\sigma = 0.02$, Control)	95.96%	77.26%	100.00%	10.00% **(Collapsed)**
Normal ($\sigma = 0.10$, Control)	94.67%	74.42%	77.68%	83.44%

Vision models (ResNet-18 and VGG-16) remain stable across the entire 50-fold range ($\Delta t = 100$ to 5000). ResNet-18 maintains between 76.12% and 76.92% accuracy while reaching 90.71%–92.11% sparsity. Similarly, VGG-16 stays within 83.36%–84.82% accuracy, with more frequent pruning yielding slightly higher final sparsity (93.30% at $\Delta t = 100$ vs. 88.35% at $\Delta t = 5000$). In contrast, MiniBERT is more sensitive to update frequency. Pruning too often ($\Delta t = 100$, over 10 times per epoch) leads to severe degradation (51.91% accuracy, 97.11% sparsity), as attention weights are cut before the network can recover. At moderate intervals ($\Delta t \geq 500$, at most 2 prunes per epoch), the model has enough training steps between cuts to stabilize representations, consistently reaching 80.35%–81.19% accuracy. We therefore use $\Delta t = 500$ as a practical default across modalities.

Table 3: Ablation study on the effect of pruning interval (Δt) across architectures (mean $\pm$ std across 3 random seeds). Vision architectures exhibit strong stability across all frequencies, whereas transformer models require moderate intervals ($\Delta t \geq 500$) to prevent rapid capacity collapse.

	ResNet-18 (CIFAR-10)		**VGG-16** (CIFAR-10)		**MiniBERT** (SST-2)	
Prune Interval (Δt)	**Sparsity (%)**	**Test Acc (%)**	**Sparsity (%)**	**Test Acc (%)**	**Sparsity (%)**	**Test Acc (%)**
$\Delta t = 100$	$92.11 \pm 0.15\%$	$76.12 \pm 0.40\%$	$93.30 \pm 0.18\%$	$84.82 \pm 0.20\%$	$97.11 \pm 0.12\%$	$51.91 \pm 1.41\%$
$\Delta t = 250$	$91.64 \pm 0.03\%$	$76.92 \pm 0.52\%$	$91.58 \pm 0.46\%$	$83.36 \pm 0.63\%$	$96.58 \pm 0.50\%$	$72.06 \pm 13.41\%$
$\Delta t = 500$ (Default)	$91.41 \pm 0.08\%$	$76.76 \pm 0.16\%$	$90.42 \pm 0.93\%$	$84.20 \pm 0.73\%$	$90.07 \pm 2.99\%$	$80.35 \pm 0.69\%$
$\Delta t = 1000$	$91.13 \pm 0.04\%$	$76.21 \pm 0.40\%$	$89.62 \pm 0.72\%$	$84.02 \pm 2.11\%$	$85.67 \pm 1.01\%$	$80.70 \pm 0.19\%$
$\Delta t = 1500$	$90.97 \pm 0.05\%$	$76.84 \pm 0.29\%$	$89.37 \pm 0.80\%$	$84.53 \pm 0.35\%$	$84.65 \pm 0.73\%$	$80.85 \pm 0.41\%$
$\Delta t = 2000$	$90.71 \pm 0.01\%$	$76.80 \pm 0.12\%$	$88.84 \pm 0.75\%$	$84.50 \pm 0.87\%$	$80.77 \pm 3.70\%$	$80.73 \pm 0.28\%$
$\Delta t = 5000$	$90.87 \pm 0.03\%$	$76.86 \pm 0.40\%$	$88.35 \pm 0.64\%$	$83.62 \pm 0.64\%$	$74.91 \pm 4.41\%$	$81.19 \pm 0.09\%$

5 Discussion

Across five architectures and four benchmarks, DADP matches or exceeds static and dynamic baselines at high sparsities (e.g., 72.28% accuracy at 99.35% sparsity on ResNet-18 and 80.85% accuracy at 96% sparsity on MiniBERT). Rather than enforcing rigid layer budgets, a single global threshold τ allocates parameters according to network information flow. This allows vision models to preserve early convolutional filters while heavily compressing deep layers, whereas sequence models retain late classification heads. Moreover, tracking gradient-activation products maintains unimodal, zero-centered weight distributions and high effective rank, effectively avoiding both the bimodal cutoffs seen in magnitude pruning and latent representation collapse.

Our ablations confirm that DADP remains invariant across standard variance-preserving initializations (Kaiming, Xavier, Orthogonal) and stable across a $50\times$ range of pruning intervals ($\Delta t \in [100, 5000]$) in vision backbones. Still, a practical limitation is identifying a valid threshold τ: setting it too aggressively can trigger network collapse. This is especially true for sequence models like MiniBERT when pruned too frequently ($\Delta t = 100$) or for sequential architectures like VGG-16 under unscaled initializations. Tracking I_{ij}^{l} also introduces transient memory overhead during the accumulation window. Due to compute limits, our evaluation focused on compact backbones and standard benchmarks (CIFAR-10, SST-2). Scaling DADP to deeper models (ResNet-50, mBERT) on larger datasets like TinyImageNet and developing structured block-pruning for hardware speedups remains an important direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools for writing assistance, code optimization, formatting LaTeX tables and writing mathematical proofs. We have not used generative AI tools for formulating the underlying scientific hypotheses, conceptualizing the core Hebbian activity-dependent pruning mechanism, or generating synthetic or artificial experimental data. All empirical results reflect actual training runs on standard benchmarks.

We have reviewed and verified all AI-assisted work. Specifically, all code was tested, debugged, and validated against baseline implementations. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure complete reproducibility of all theoretical and empirical results, we have organized the relevant documentation across the manuscript, appendix, and supplementary materials:

- **Theoretical Proofs:** Formal derivations, assumptions, and complete proofs for Theorem 1 and Theorem 2 are provided in Appendix A.
- **Architectures and Hyperparameters:** Detailed layer specifications, training epochs, learning rates, batch sizes, pruning intervals (ΔT), and threshold sweeps (τ) are detailed in Section 3.3 and summarized in Table 4 (Appendix C).
- **Baseline Adaptation Details:** Implementation specifics and training schedules for Magnitude Pruning, SNIP, and RigL under matching compute budgets are described in Appendix C.2.
- **Statistical Significance:** Full empirical benchmark tables reporting mean $\pm$ standard deviation across multiple random seeds are provided in Appendix D.
- **Code and Implementation:** Anonymized source code containing the core DADP pruning algorithm, model architectures, and evaluation scripts is included directly in the supplementary package submitted alongside this manuscript. Furthermore, the complete, clean codebase with end-to-end training and benchmark evaluation pipelines will be open-sourced upon publication.

ETHICS STATEMENT

This work introduces an algorithmic method for deep neural network compression. All datasets used in this study (MNIST, CIFAR-10, CoNLL-2003, and SST-2) are standard, publicly available academic benchmarks. This research does not involve human subjects, personal data privacy concerns, or potential dual-use biological or security harms. By lowering computational and energy demands during both training and inference, network pruning supports more sustainable and environmentally accessible machine learning.

ACKNOWLEDGMENTS

We gratefully acknowledge Google Colab and Kaggle for providing free cloud GPU compute resources (NVIDIA T4), which enabled the execution of all training, pruning, and empirical evaluation experiments presented in this work.

REFERENCES

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning (ICML)*, pp. 254–263, 2018.

Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Joseph Bingham, Shizhen Zhao, and Daniel Alon. Fine-Pruning: A biologically inspired algorithm for personalization of ML models. *arXiv preprint arXiv:2502.14886*, 2025.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006. ISBN 978-0-387-31073-2.

Yves Chauvin. A back-propagation algorithm with optimal use of hidden units. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1989.

Aleksandr Dekhovich, David M. J. Tax, Marcel H. F. Sluiter, and Miguel A. Bessa. Neural network relief: a pruning algorithm based on neural activity. *arXiv preprint arXiv:2308.02060*, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 4171–4186, 2019.

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)*, pp. 2943–2952, 2020.

Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. DepGraph: Towards any structural pruning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Travis E Faust, Georgia Gunner, and Dorothy P Schafer. Mechanisms governing activity-dependent synaptic pruning in the developing mammalian CNS. *Nature Reviews Neuroscience*, 22(11): 657–673, 2021.

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

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one shot. In *International Conference on Machine Learning (ICML)*, 2023.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 249–256, 2010.

Dongcheng Han, Tielin Zhang, and Bo Xu. Developmental plasticity-inspired adaptive pruning for deep spiking and artificial neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(10):6890–6904, 2024.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, pp. 1135–1143, 2015.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *International Conference on Learning Representations (ICLR)*, 2016.

Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 164–171, 1993.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Donald Olding Hebb. *The Organization of Behavior: A Neuropsychological Theory*. John Wiley & Sons, 1949.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Peter R Huttenlocher. *Neural Plasticity: The Effects of Environment on the Development of the Cerebral Cortex*. Harvard University Press, 2002.

Peter R. Huttenlocher, Ch. de Courten, L. J. Garey, and H. Van der Loos. Synaptic density in human frontal cortex–developmental changes and effects of aging. *Brain Research*, 163(2):195–205, 1979.

Ehud D Karnin. A simple procedure for pruning back-propagation trained neural networks. *IEEE Transactions on Neural Networks*, 1(2):239–242, 1990.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 260–270, 2016.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in Neural Information Processing Systems (NeurIPS)*, 2, 1989.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. SNIP: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations (ICLR)*, 2019.

Takayasu Mikuni, Naofumi Uesaka, Hiroyuki Okuno, Hirokazu Hirai, Karl Deisseroth, Haruhiko Bito, and Masanobu Kano. Arc/Arg3.1 is a postsynaptic mediator of activity-dependent synapse elimination in the developing cerebellum. *Neuron*, 78(6):1024–1035, 2013.

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

Michael C Mozer and Paul Smolensky. Using relevance to reduce network size automatically. *Connection Science*, 1(1):3–16, 1989.

Russell Reed. Pruning algorithms-a survey. *IEEE Transactions on Neural Networks*, 4(5):740–747, 1993.

Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning (CoNLL)*, pp. 142–147, 2003.

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

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. In *International Conference on Machine Learning (ICML)*, 2025.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1631–1642, 2013.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for LLMs. In *International Conference on Learning Representations (ICLR)*, 2024.

Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

Hidenori Tanaka, Aran Nayebi, Niru Maheswaranathan, Lane McIntosh, Stephen Baccus, and Surya Ganguli. From deep learning to mechanistic understanding in neuroscience: the structure of retinal prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8535–8545, 2019.

Hidenori Tanaka, Daniel Kunin, Daniel L. Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:6377–6389, 2020.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations (ICLR)*, 2020.

Andreas S Weigend, David E Rumelhart, and Bernardo A Huberman. Generalization by weight-elimination with application to forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1991.

Masahiro Yasuda, Sivapratha Nagappan-Chettiar, Erin M Johnson-Venkatesh, and Hisashi Umemori. An activity-dependent determinant of synapse elimination in the mammalian brain. *Neuron*, 109 (8):1333–1349, 2021.

A MATHEMATICAL PROOFS

In this section, we present the formal derivations and proofs for Theorem 1 and Theorem 2.

A.1 PROOF OF THEOREM 1: FIRST-ORDER TAYLOR EXPANSION OF LOSS SENSITIVITY

Theorem. *The activation-derivative metric* $I_{ij}^l = \left| a_i^{l-1} \cdot \frac{\partial \mathcal{L}}{\partial y_j^l} \right|$ *is equivalent to the magnitude of the first-order Taylor expansion of loss change* $\Delta\mathcal{L}$ *resulting from setting connection* w_{ij}^l *to zero, normalized per unit weight magnitude.*

Proof. Let $\mathcal{L}(\Theta)$ be a twice-differentiable loss function over parameters Θ. Consider an active connection $w_{ij}^l \in \mathbb{R}$ linking input neuron i at layer $l-1$ with pre-activation $y_j^l = \sum_k w_{kj}^l a_k^{l-1} + b_j^l$ at layer l.

Pruning weight w_{ij}^l to zero corresponds to setting $w_{ij}^{l,\text{pruned}} = 0$, which induces a weight perturbation $\Delta w_{ij}^l = 0 - w_{ij}^l = -w_{ij}^l$.

Expanding the perturbation of loss $\mathcal{L}$ around the current weight value w_{ij}^l using a first-order Taylor series yields:

$$\Delta\mathcal{L} = \mathcal{L}(w_{ij}^l + \Delta w_{ij}^l) - \mathcal{L}(w_{ij}^l) \approx \frac{\partial \mathcal{L}}{\partial w_{ij}^l} \Delta w_{ij}^l = -\frac{\partial \mathcal{L}}{\partial w_{ij}^l} w_{ij}^l \tag{5}$$

By the multivariable chain rule, the partial derivative of loss with respect to weight w_{ij}^l decomposes as:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^l} = \frac{\partial \mathcal{L}}{\partial y_j^l} \cdot \frac{\partial y_j^l}{\partial w_{ij}^l} \tag{6}$$

Since $y_j^l = w_{ij}^l a_i^{l-1} + \sum_{k \neq i} w_{kj}^l a_k^{l-1} + b_j^l$, the local partial derivative is:

$$\frac{\partial y_j^l}{\partial w_{ij}^l} = a_i^{l-1} \tag{7}$$

Substituting Equation equation 7 into Equation equation 6 gives:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^l} = \frac{\partial \mathcal{L}}{\partial y_j^l} \cdot a_i^{l-1} \tag{8}$$

Substituting Equation equation 8 back into the Taylor expansion in Equation equation 5 produces:

$$\Delta\mathcal{L} \approx -w_{ij}^l \left(a_i^{l-1} \cdot \frac{\partial \mathcal{L}}{\partial y_j^l} \right) \tag{9}$$

Dividing by the absolute magnitude of active weight $|w_{ij}^l|$ to obtain the scale-invariant loss sensitivity per unit weight magnitude gives:

$$\left| \frac{\Delta\mathcal{L}}{w_{ij}^l} \right| \approx \left| a_i^{l-1} \cdot \frac{\partial \mathcal{L}}{\partial y_j^l} \right| = I_{ij}^l \tag{10}$$

This completes the proof. ■ □

A.2 ANALYSIS OF THEOREM 2: TRAINING-DRIVEN SPARSITY EQUILIBRIUM & GENERALIZATION CAPACITY BOUNDARY

Theorem. *Given a global threshold* τ*, DADP importance scores* $I_{ij}^l = \mathbb{E}[|a_i^{l-1} \cdot \frac{\partial \mathcal{L}_{train}}{\partial y_j^l}|]$ *stabilize surviving pathways above* τ*, causing the structural pruning rate to asymptote (*$\Delta S \to 0$*) into an organic training sparsity equilibrium* $S^*(\tau) < 100\%$*. However, because* I_{ij}^l *is computed on training set loss sensitivity, setting* τ *near the critical capacity boundary* τ_{crit} *preserves high training accuracy while causing test loss to increase and test generalization to degrade.*

Proof. Let $\mathcal{P}^*_{\text{train}} = \{(i,j,l) \mid \text{connection } w^l_{ij} \text{ is essential for training loss minimization}\}$ denote the active subnetwork required to fit the training distribution.

Assume for contradiction that as training proceeds ($t \to \infty$), global sparsity reaches total structural collapse $S(t) \to 100\%$, implying that all connections $(i,j,l) \in \mathcal{P}^*_{\text{train}}$ are pruned because $I^l_{ij}(t) < \tau$. If all essential training connections in $\mathcal{P}^*_{\text{train}}$ are pruned, the information flow to output logits collapses, causing training error to spike ($\mathcal{L}_{\text{train}}(t) \to \infty$).

Under standard gradient backpropagation, as training loss $\mathcal{L}_{\text{train}} \to \infty$, the output error gradient $\frac{\partial \mathcal{L}_{\text{train}}}{\partial y^L_j}$ scales proportionally:

$$\left\| \frac{\partial \mathcal{L}_{\text{train}}}{\partial y^L} \right\| \to \infty \tag{11}$$

Backpropagating error gradients through surviving active pathways yields:

$$\frac{\partial \mathcal{L}_{\text{train}}}{\partial y^l_j} = \sum_k \left(\frac{\partial \mathcal{L}_{\text{train}}}{\partial y^{l+1}_k} \cdot w^{l+1}_{jk} \cdot \sigma'(y^l_j) \right) \tag{12}$$

The gradient magnitude $\left| \frac{\partial \mathcal{L}_{\text{train}}}{\partial y^l_j} \right|$ increases monotonically.

For essential active connections $(i,j,l) \in \mathcal{P}^*_{\text{train}}$, the expectation score $I^l_{ij}(t+1)$ satisfies:

$$I^l_{ij}(t+1) = \frac{1}{\Delta T} \sum_{k=0}^{\Delta T - 1} \left| a^{l-1}_i \cdot \frac{\partial \mathcal{L}_{\text{train}}}{\partial y^l_j} \right| \geq \tau \tag{13}$$

Whenever $I^l_{ij}(t+1) \geq \tau$, the connection remains active: $M^l_{ij}(t+1) = 1$. This active connection prevents training loss $\mathcal{L}_{\text{train}}$ from diverging, stabilizing I^l_{ij} at a stationary equilibrium score $I^* \geq \tau$ and establishing a structural sparsity asymptote ($\Delta S \to 0$).

Generalization Capacity Boundary. Importantly, because the importance score I^l_{ij} depends on training set gradients $\frac{\partial \mathcal{L}_{\text{train}}}{\partial y^l_j}$, the structural equilibrium $S^*(\tau)$ is governed strictly by training set fitting. As the network becomes extremely sparse (e.g., $> 98.0\%$ sparsity in 100-epoch limit tests), the surviving active weights fit the training data near perfectly (99.53% train accuracy), keeping $I^l_{ij} \geq \tau$. However, the loss of representation capacity causes the network to overfit, leading to an increase in test loss ($0.0945 \to 0.2277$) and a slight decay in test accuracy ($97.69\% \to 96.94\%$). Pushing $\tau \geq \tau_{\text{crit}}$ (e.g., $\tau =$ 1e-4) forces the model past this capacity boundary into catastrophic classification collapse (77.11%). ■ □

B DADP ALGORITHMIC PROCEDURE

We provide the complete procedural workflow for Dynamic Activity-Dependent Pruning (DADP) in Algorithm 1. In DADP, synaptic importance accumulators track the elementwise product of forward activations and backpropagated gradients across each mini-batch. At periodic step intervals ΔT, temporal averages are computed and thresholded against τ to permanently prune redundant pathways without regrowth.

Algorithm 1 Dynamic Activity-Dependent Pruning (DADP)

Require: Training dataset $\mathcal{D}$, initial weights $W = \{W^l\}_{l=1}^{L}$, learning rate η, pruning interval ΔT, global threshold τ, total steps T.
Ensure: Sparsified model parameters $W \odot M$.

```
Initialize masks M_ij^l ← 1 for all layers l and connections (i, j)
Initialize importance accumulators A_ij^l ← 0
for step t = 1 to T do
    Sample mini-batch (X, y) ~ D
    Forward pass with masked parameters: y^l = (W^l ⊙ M^l) a^{l-1} + b^l
    Compute loss L and backpropagate error gradients ∂L/∂y_j^l
    Accumulate local sensitivity: A_ij^l ← A_ij^l + | a_i^{l-1} · ∂L/∂y_j^l |
    Update weights with optimizer: W^l ← OptimizerStep(W^l, ∇_{W^l} L)
    Enforce mask: W^l ← W^l ⊙ M^l
    if t ≡ 0 (mod ΔT) then
        Compute temporal average: I_ij^l ← A_ij^l / ΔT
        Update masks: M_ij^l ← M_ij^l · 𝕀(I_ij^l ≥ τ)
        Permanently prune weights: W^l ← W^l ⊙ M^l
        Reset accumulators: A_ij^l ← 0
    end if
end for
return W ⊙ M
```

C Network Architectures and Training Setup

We provide detailed structural specifications and training configurations for all five neural network architectures evaluated in this work.

C.1 Architectural Details

1. **MLP (MNIST)**: A fully connected feedforward network designed for 28×28 grayscale digit classification (`BaselineMLP` / `HebbianMLP`). It consists of an input flattening layer (784 dimensions), followed by two hidden linear layers of dimension 512 each with ReLU activations (`fc1`: $784 \to 512$, `fc2`: $512 \to 512$). The final linear classification head (`fc3`) maps from 512 dimensions to 10 class logits ($669,706$ total parameters).

2. **VGG-16 (CIFAR-10)**: A 16-layer convolutional architecture adapted for $32 \times 32 \times 3$ images (`BaselineVGG16` / `HebbianVGG16`). The feature extractor comprises 13 convolutional layers arranged into 5 blocks ($[2, 2, 3, 3, 3]$ conv layers per block with channel depths $64, 128, 256, 512, 512$). Each convolution uses a 3×3 kernel, stride 1, and padding 1, followed by Batch Normalization and ReLU. Max-pooling (2×2, stride 2) is applied after blocks 1, 2, 3, and 4. Spatial feature maps are condensed via `AdaptiveAvgPool2d((1, 1))`. The classifier consists of three dense layers ($512 \to 512 \to 10$) with dropout ($p = 0.5$), yielding $15,253,578$ total parameters.

3. **ResNet-18 (CIFAR-10)**: A residual network tailored for CIFAR-10 image classification (`NativeResNet18` / `get_resnet18`). It features an initial convolutional layer (64 filters, 7×7 kernel, stride 2, padding 3) with Batch Normalization, ReLU, and 3×3 max-pooling, followed by 4 residual stages containing 2 BasicBlocks each ($[64, 128, 256, 512]$ feature channels). Residual downsampling shortcuts utilize 1×1 convolutions with stride 2 and Batch Normalization. Global Average Pooling condenses spatial maps to 512 dimensions, followed by a linear projection to 10 classes ($11,181,642$ total parameters).

4. **BiLSTM-CRF (CoNLL-2003)**: A sequence tagging model for Named Entity Recognition (`BiLSTM_CRF`). Input tokens are mapped via a trainable 128-dimensional embedding layer (vocabulary size $5,000$), followed by a 1-layer Bidirectional LSTM with hidden state dimension $h = 64$ per direction (128-dimensional concatenated output per step). A linear projection (`hidden2tag`) maps hidden states to emission scores across 9 entity classes ($128 \to 9$), followed by a Linear-Chain Conditional Random Field (`CRF`) layer with a 9×9 learnable transition matrix trained via Negative Log-Likelihood ($740,588$ total parameters).

5. **MiniBERT (SST-2)**: HuggingFace's pretrained `prajjwal1/bert-tiny` architecture fine-tuned for binary sentiment classification. It comprises a vocabulary of $30,522$ tokens (128-dim embeddings), 2 Transformer encoder layers, hidden dimension $d = 128$, intermediate feedforward dimension $d_{\text{ff}} = 512$, and 2 self-attention heads per layer (maximum sequence length 128). A pooled linear classification head maps representations to 2 sentiment logits ($4,386,178$ total parameters).

Table 4: Hyperparameter configurations and training specifications across all evaluated architectures.

Hyperparameter	**MLP**	**VGG-16**	**ResNet-18**	**BiLSTM-CRF**	**MiniBERT**
Dataset	MNIST	CIFAR-10	CIFAR-10	CoNLL-2003	SST-2
Task Modality	Vision (Digits)	Vision (Objects)	Vision (Objects)	Sequence (NER)	NLP (Sentiment)
Total Parameters	667,146	14,728,266	11,173,962	1,514,841	4,386,178
Optimizer	Adam	Adam	Adam	Adam	Adam
Learning Rate (η)	1e-3	1e-3	1e-3	1e-3	2e-5
Weight Decay	1e-4	1e-4	1e-4	1e-4	1e-2
Batch Size	64	64	64	64	64
Training Epochs	20	20	20	20	20
Loss Function	Cross-Entropy	Cross-Entropy	Cross-Entropy	CRF NLL Loss	Cross-Entropy
Prune Interval (ΔT)	500 steps	500 steps	500 steps	500 steps	500 steps
Threshold Sweep (τ)	[1e-6, 1e-4]	[1e-6, 1e-4]	[1e-6, 5e-4]	[1e-6, 3e-4]	[1e-6, 5e-6]
Precision	FP16 (AMP)	FP16 (AMP)	FP16 (AMP)	FP32	FP16 (AMP)
Hardware Accelerator	NVIDIA T4	NVIDIA T4	NVIDIA T4	NVIDIA T4	NVIDIA T4

C.2 Baseline Pruning Implementation Details

To ensure a fair comparison, all models were trained for a total of 20 epochs under matching computational budgets. We adapted the baseline pruning methods to fit this schedule:

- **Magnitude Pruning:** We train the model as a dense network for 17 epochs. At epoch 17, we prune the smallest weights to reach the target sparsity. We then fine-tune the sparse model for the remaining 3 epochs.
- **SNIP:** We calculate the pruning mask at epoch 0 before training begins using a single batch of data. Once the lowest-scoring weights are pruned, the mask stays fixed for all 20 training epochs.
- **RigL:** We start training from a random sparse mask at epoch 0. Every 100 steps, we drop small weights and regrow connections with large gradients. To keep training fast within our 20-epoch limit, we gradually slow down the regrowth rate over the first 16 epochs using a cosine curve, then freeze the mask for the final 4 epochs for fine-tuning.

D FULL EMPIRICAL BENCHMARK TABLES

In this section, we provide complete benchmark evaluation tables across all models, datasets, pruning thresholds, target sparsities, and weight initialization strategies. DADP results report mean $\pm$ standard deviation across 3 random seeds $(42, 512, 1729)$ from statistical sweeps.

D.1 MLP ON MNIST

Table 5: Complete empirical evaluation for MLP on MNIST across pruning methods and thresholds (DADP reports mean $\pm$ std across 3 random seeds).

Method	Configuration / Threshold	Final Sparsity (%)	Final Test Acc (%)	Peak Test Acc (%)
Dense Baseline	Unpruned (0%)	0.00%	98.39%	98.44%
DADP (Ours)	$\tau =$ 1e-6	**84.90 $\pm$ 0.56%**	**97.91 $\pm$ 0.12%**	**98.18 $\pm$ 0.02%**
DADP (Ours)	$\tau =$ 5e-6	**94.60 $\pm$ 0.14%**	**97.55 $\pm$ 0.09%**	**98.02 $\pm$ 0.03%**
DADP (Ours)	$\tau =$ 1e-5	**96.19 $\pm$ 0.11%**	**97.38 $\pm$ 0.15%**	**97.91 $\pm$ 0.08%**
DADP (Ours)	$\tau =$ 5e-5	**98.35 $\pm$ 0.04%**	**96.69 $\pm$ 0.04%**	**96.92 $\pm$ 0.05%**
DADP (Ours)	$\tau =$ 1e-5 (100 Epoch Limit Test)	**98.00%**	**96.94%**	**97.69%**
DADP (Ours)	$\tau =$ 1e-4 (Over-threshold)	**99.81%**	**77.11%**	**77.30%**
Magnitude Pruning	Target 70.0%	70.00%	98.58%	98.58%
Magnitude Pruning	Target 80.0%	80.00%	98.63%	98.65%
Magnitude Pruning	Target 90.0%	90.00%	98.49%	98.49%
Magnitude Pruning	Target 95.0%	95.00%	98.05%	98.38%
RigL	Target 70.0%	70.00%	98.18%	98.25%
RigL	Target 80.0%	80.00%	97.97%	98.21%
RigL	Target 90.0%	90.00%	97.81%	98.00%
RigL	Target 95.0%	95.00%	97.65%	97.94%
SNIP	Target 70.0%	70.00%	98.23%	98.40%
SNIP	Target 80.0%	80.00%	97.85%	98.29%
SNIP	Target 90.0%	90.00%	97.89%	98.18%
SNIP	Target 95.0%	95.00%	97.82%	97.94%

D.2 RESNET-18 ON CIFAR-10

Table 6: Complete empirical evaluation for ResNet-18 on CIFAR-10 across pruning methods and thresholds (DADP reports mean $\pm$ std across 3 random seeds).

Method	Configuration / Threshold	Final Sparsity (%)	Final Test Acc (%)	Peak Test Acc (%)
Dense Baseline	Unpruned (0%)	0.00%	76.06%	77.44%
DADP (Ours)	$\tau =$ 1e-6	**84.65 $\pm$ 0.21%**	**76.72 $\pm$ 0.26%**	**76.97 $\pm$ 0.25%**
DADP (Ours)	$\tau =$ 5e-6	**89.66 $\pm$ 0.10%**	**76.64 $\pm$ 0.75%**	**77.36 $\pm$ 0.17%**
DADP (Ours)	$\tau =$ 1e-5	**91.41 $\pm$ 0.08%**	**76.76 $\pm$ 0.16%**	**77.25 $\pm$ 0.24%**
DADP (Ours)	$\tau =$ 5e-5	**95.63 $\pm$ 0.05%**	**76.15 $\pm$ 0.34%**	**77.08 $\pm$ 0.16%**
DADP (Ours)	$\tau =$ 1e-4	**96.92 $\pm$ 0.01%**	**76.03 $\pm$ 0.61%**	**76.80 $\pm$ 0.21%**
DADP (Ours)	$\tau =$ 5e-4 (High Sparsity)	**99.35 $\pm$ 0.03%**	**72.28 $\pm$ 0.28%**	**72.90 $\pm$ 0.24%**
Magnitude Pruning	Target 70.0%	70.00%	77.41%	77.41%
Magnitude Pruning	Target 80.0%	80.00%	77.51%	77.53%
Magnitude Pruning	Target 90.0%	90.00%	77.07%	77.32%
Magnitude Pruning	Target 95.0%	95.00%	77.18%	77.18%
Magnitude Pruning	Target 99.0%	99.00%	66.42%	77.21%
RigL	Target 70.0%	70.00%	74.14%	74.76%
RigL	Target 80.0%	80.00%	72.96%	74.01%
RigL	Target 90.0%	90.00%	71.53%	72.34%
RigL	Target 95.0%	95.00%	69.60%	70.45%
RigL	Target 99.0%	99.00%	63.85%	64.36%
SNIP	Target 70.0%	70.00%	75.99%	76.94%
SNIP	Target 80.0%	80.00%	75.92%	76.83%
SNIP	Target 90.0%	90.00%	75.21%	76.10%
SNIP	Target 95.0%	95.00%	75.49%	75.49%
SNIP	Target 99.0%	99.00%	71.44%	72.30%

D.3 BiLSTM-CRF on CoNLL-2003

Table 7: Complete empirical evaluation for BiLSTM-CRF on CoNLL-2003 (DADP reports mean ± std across 3 random seeds).

Method	Configuration / Threshold	Final Sparsity (%)	Final Test F1 (%)	Peak Test F1 (%)
Dense Baseline	Unpruned (0%)	0.00%	85.16%	85.16%
DADP (Ours)	τ = 1e-6	**12.10 ± 3.16%**	**93.77 ± 0.12%**	**93.94 ± 0.13%**
DADP (Ours)	τ = 5e-6	**26.19 ± 3.80%**	**93.82 ± 0.06%**	**93.96 ± 0.15%**
DADP (Ours)	τ = 1e-5	**33.57 ± 4.06%**	**93.82 ± 0.14%**	**93.95 ± 0.13%**
DADP (Ours)	τ = 5e-5	**95.00 ± 3.60%**	**84.28 ± 1.34%**	**93.82 ± 0.23%**
DADP (Ours)	τ = 1e-4	**95.09 ± 0.98%**	**87.99 ± 1.73%**	**93.25 ± 0.05%**
DADP (Ours)	τ = 3e-4 (Over-threshold)	**100.00%**	**82.82%**	**92.34%**
Magnitude Pruning	Target 70.0%	70.00%	93.87%	94.18%
Magnitude Pruning	Target 80.0%	80.00%	93.88%	94.47%
Magnitude Pruning	Target 90.0%	90.00%	94.06%	94.18%
Magnitude Pruning	Target 95.0%	95.00%	93.88%	93.88%
RigL	Target 70.0%	70.00%	93.26%	94.06%
RigL	Target 80.0%	80.00%	93.44%	94.00%
RigL	Target 90.0%	90.00%	93.78%	94.32%
RigL	Target 95.0%	95.00%	93.64%	93.78%
SNIP	Target 70.0%	70.00%	93.50%	94.28%
SNIP	Target 80.0%	80.00%	93.37%	93.77%
SNIP	Target 90.0%	90.00%	93.19%	94.24%
SNIP	Target 95.0%	95.00%	92.60%	94.02%

D.4 MiniBERT on SST-2

Table 8: Complete empirical evaluation for MiniBERT on SST-2 across pruning methods and thresholds (DADP reports mean ± std across 3 random seeds).

Method	Configuration / Threshold	Final Sparsity (%)	Final Test Acc (%)	Peak Test Acc (%)
Dense Baseline	Unpruned (0%)	0.00%	80.62%	82.68%
DADP (Ours)	τ = 1e-6	**71.52 ± 4.93%**	**79.01 ± 0.57%**	**82.19 ± 0.39%**
DADP (Ours)	τ = 2e-6	**91.69 ± 2.03%**	**78.67 ± 0.66%**	**81.57 ± 0.30%**
DADP (Ours)	τ = 3e-6	**96.03 ± 0.79%**	**80.85 ± 1.06%**	**82.11 ± 0.37%**
DADP (Ours)	τ = 4e-6 (Over-threshold)	**96.87 ± 0.16%**	**64.18 ± 12.70%**	**64.22 ± 12.76%**
DADP (Ours)	τ = 5e-6 (Over-threshold)	**96.96%**	**50.92%**	**50.92%**
Magnitude Pruning	Target 70.0%	70.00%	50.92%	82.57%
Magnitude Pruning	Target 80.0%	80.00%	50.92%	82.34%
Magnitude Pruning	Target 90.0%	90.00%	50.92%	83.03%
Magnitude Pruning	Target 95.0%	95.00%	50.92%	82.34%
RigL	Target 70.0%	70.00%	80.62%	80.62%
RigL	Target 80.0%	80.00%	80.73%	81.42%
RigL	Target 90.0%	90.00%	80.62%	81.77%
RigL	Target 95.0%	95.00%	81.31%	81.31%
SNIP	Target 70.0%	70.00%	81.54%	82.22%
SNIP	Target 80.0%	80.00%	81.54%	82.00%
SNIP	Target 90.0%	90.00%	82.68%	82.80%
SNIP	Target 95.0%	95.00%	81.65%	82.22%

E Complete VGG-16 Empirical Evaluation & Visual Profiles

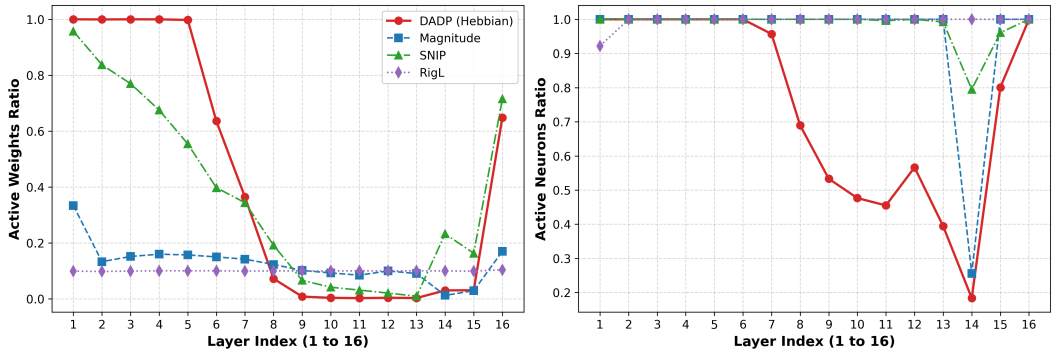
To provide a comprehensive examination of DADP's behavior on large feedforward convolutional networks, this section consolidates all empirical benchmarks, layer-wise sparsity allocations, information-theoretic representation metrics, and weight distribution spectra for VGG-16 on CIFAR-10.

Table 9: Complete empirical evaluation for VGG-16 on CIFAR-10 across pruning methods and thresholds (DADP reports mean ± std across 3 random seeds).

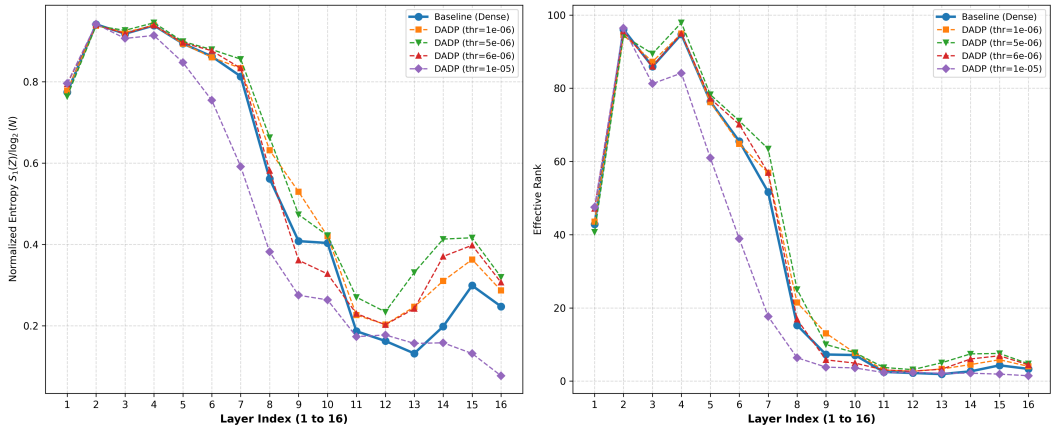
Method	Configuration / Threshold	Final Sparsity (%)	Final Test Acc (%)	Peak Test Acc (%)
Dense Baseline	Unpruned (0%)	0.00%	85.21%	85.21%
DADP (Ours)	τ = 1e-6	**74.26 ± 1.25%**	**85.09 ± 0.17%**	**85.11 ± 0.16%**
DADP (Ours)	τ = 5e-6	**89.77 ± 0.08%**	**84.70 ± 0.17%**	**85.47 ± 0.34%**
DADP (Ours)	τ = 6e-6	**91.65 ± 0.67%**	**82.65 ± 0.65%**	**84.29 ± 0.53%**
DADP (Ours)	τ = 6.5e-6	**91.86 ± 0.54%**	**83.22 ± 0.00%**	**83.22 ± 0.00%**
DADP (Ours)	τ = 6.7e-6	**92.16 ± 0.28%**	**83.06 ± 2.01%**	**84.02 ± 1.36%**
DADP (Ours)	τ = 1e-5 (Over-threshold)	**96.50%**	**10.00%**	**18.84%**
DADP (Ours)	τ = 1e-4 (Over-threshold)	**100.00%**	**10.00%**	**10.00%**
Magnitude Pruning	Target 70.0%	70.00%	86.03%	86.22%
Magnitude Pruning	Target 80.0%	80.00%	85.75%	85.75%
Magnitude Pruning	Target 90.0%	90.00%	85.56%	85.56%
Magnitude Pruning	Target 95.0%	95.00%	82.83%	84.49%
RigL	Target 70.0%	70.00%	83.51%	84.07%
RigL	Target 80.0%	80.00%	81.20%	83.67%
RigL	Target 90.0%	90.00%	82.38%	82.38%
RigL	Target 95.0%	95.00%	80.23%	80.58%
SNIP	Target 70.0%	70.00%	85.39%	85.90%
SNIP	Target 80.0%	80.00%	84.83%	85.71%
SNIP	Target 90.0%	90.00%	84.65%	85.11%
SNIP	Target 95.0%	95.03%	14.06%	23.44%

E.1 VGG-16 Numerical Benchmarks

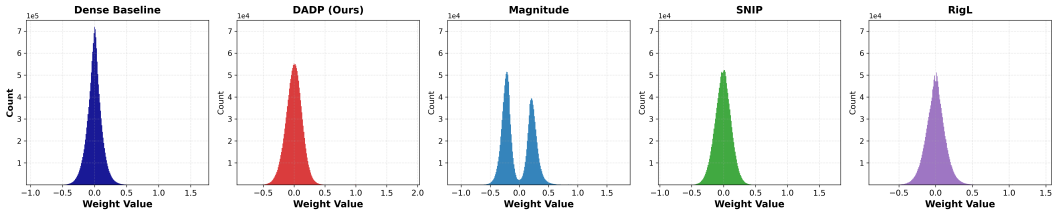
E.2 VGG-16 Visual Profiles & Representations



(a) Layer-Wise Sparsity Profile (90% Sparsity)



(b) Information-Theoretic Representation Quality (Matrix Entropy & Effective Rank)



(c) Global Active Weight Value Distributions across Pruning Methods

Figure 5: Comprehensive visual evaluation of VGG-16 on CIFAR-10. (a) DADP preserves classification accuracy up to 92.16% global sparsity. (b) Layer-wise profile shows early spatial filter lock alongside heavy compression in deep linear projection layers (`classifier.0`). (c) Latent representation quality across all 16 layers confirms that DADP maintains normalized Shannon entropy ($S_{\text{norm}} \geq 0.85$) on par with dense unpruned features. (d) DADP preserves a smooth, zero-centered parameter distribution, avoiding artificial bimodal gaps around zero.

F Detailed Channel-Level Structured Compression Tables

In this section, we present complete layer-by-layer physical structured compression shapes for VGG-16 (at $\sim 90\%$ global sparsity) and ResNet-18 (at $\sim 95\%$ global sparsity) across all evaluated pruning methods (DADP, SNIP, Magnitude, and RigL). Physical structured pruning deletes completely dead channels and neurons (where 100% of constituent weights are inactive), compressing tensor dimensions directly on disk.

F.1 VGG-16 on CIFAR-10 ($\sim 90\%$ Global Sparsity)

Table 10: Layer-by-layer physical structured compression shapes for VGG-16 on CIFAR-10 at $\sim 90\%$ global sparsity across pruning methods, with the number of pruned (dead) channels/neurons in parentheses.

Layer Name	Original Shape	DADP Compressed Shape	SNIP Compressed Shape	Magnitude Compressed Shape	RigL Compressed Shape
features.0	[64, 3, 3, 3]	[64, 3, 3, 3] (0 dead)	[64, 3, 3, 3] (0 dead)	[64, 3, 3, 3] (0 dead)	[59, 3, 3, 3] (5 dead)
features.3	[64, 64, 3, 3]	[64, 64, 3, 3] (0 dead)	[64, 64, 3, 3] (0 dead)	[64, 64, 3, 3] (0 dead)	[64, 59, 3, 3] (0 dead)
features.7	[128, 64, 3, 3]	[128, 64, 3, 3] (0 dead)	[128, 64, 3, 3] (0 dead)	[128, 64, 3, 3] (0 dead)	[128, 64, 3, 3] (0 dead)
features.10	[128, 128, 3, 3]	[128, 128, 3, 3] (0 dead)	[128, 128, 3, 3] (0 dead)	[128, 128, 3, 3] (0 dead)	[128, 128, 3, 3] (0 dead)
features.14	[256, 128, 3, 3]	[256, 128, 3, 3] (0 dead)	[256, 128, 3, 3] (0 dead)	[256, 128, 3, 3] (0 dead)	[256, 128, 3, 3] (0 dead)
features.17	[256, 256, 3, 3]	[256, 256, 3, 3] (0 dead)	[256, 256, 3, 3] (0 dead)	[256, 256, 3, 3] (0 dead)	[256, 256, 3, 3] (0 dead)
features.20	[256, 256, 3, 3]	[256, 256, 3, 3] (0 dead)	[256, 256, 3, 3] (0 dead)	[256, 256, 3, 3] (0 dead)	[256, 256, 3, 3] (0 dead)
features.24	[512, 256, 3, 3]	[498, 256, 3, 3] (14 dead)	[512, 256, 3, 3] (0 dead)	[512, 256, 3, 3] (0 dead)	[512, 256, 3, 3] (0 dead)
features.27	[512, 512, 3, 3]	[444, 498, 3, 3] (68 dead)	[512, 512, 3, 3] (0 dead)	[512, 512, 3, 3] (0 dead)	[512, 512, 3, 3] (0 dead)
features.30	[512, 512, 3, 3]	[399, 444, 3, 3] (113 dead)	[512, 512, 3, 3] (0 dead)	[512, 512, 3, 3] (0 dead)	[512, 512, 3, 3] (0 dead)
features.34	[512, 512, 3, 3]	[391, 399, 3, 3] (121 dead)	[510, 512, 3, 3] (2 dead)	[512, 512, 3, 3] (0 dead)	[512, 512, 3, 3] (0 dead)
features.37	[512, 512, 3, 3]	[338, 391, 3, 3] (174 dead)	[512, 510, 3, 3] (0 dead)	[512, 512, 3, 3] (0 dead)	[512, 512, 3, 3] (0 dead)
features.40	[512, 512, 3, 3]	[293, 338, 3, 3] (219 dead)	[508, 512, 3, 3] (4 dead)	[512, 512, 3, 3] (0 dead)	[512, 512, 3, 3] (0 dead)
classifier.0	[512, 512]	[103, 293] (409 dead)	[407, 508] (105 dead)	[131, 512] (381 dead)	[512, 512] (0 dead)
classifier.3	[512, 512]	[452, 103] (60 dead)	[492, 407] (20 dead)	[512, 131] (0 dead)	[512, 512] (0 dead)
classifier.6	[10, 512]	[10, 452] (0 dead)	[10, 492] (0 dead)	[10, 512] (0 dead)	[10, 512] (0 dead)

F.2 ResNet-18 on CIFAR-10 ($\sim 95\%$ Global Sparsity)

Table 11: Layer-by-layer physical structured compression shapes for ResNet-18 on CIFAR-10 at $\sim 95\%$ global sparsity across pruning methods, with the number of pruned (dead) channels/neurons in parentheses.

Layer Name	Original Shape	DADP Compressed Shape	SNIP Compressed Shape	Magnitude Compressed Shape	RigL Compressed Shape
conv1	[64, 3, 3, 3]	[64, 3, 3, 3] (0 dead)	[64, 3, 3, 3] (0 dead)	[64, 3, 3, 3] (0 dead)	[64, 3, 3, 3] (0 dead)
layer1.0.conv1	[64, 64, 3, 3]	[64, 64, 3, 3] (0 dead)	[64, 64, 3, 3] (0 dead)	[64, 64, 3, 3] (0 dead)	[64, 64, 3, 3] (0 dead)
layer1.0.conv2	[64, 64, 3, 3]	[64, 64, 3, 3] (0 dead)	[64, 64, 3, 3] (0 dead)	[64, 64, 3, 3] (0 dead)	[64, 64, 3, 3] (0 dead)
layer1.1.conv1	[64, 64, 3, 3]	[64, 64, 3, 3] (0 dead)	[64, 64, 3, 3] (0 dead)	[64, 64, 3, 3] (0 dead)	[64, 64, 3, 3] (0 dead)
layer1.1.conv2	[64, 64, 3, 3]	[64, 64, 3, 3] (0 dead)	[64, 64, 3, 3] (0 dead)	[64, 64, 3, 3] (0 dead)	[64, 64, 3, 3] (0 dead)
layer2.0.conv1	[128, 64, 3, 3]	[128, 64, 3, 3] (0 dead)	[128, 64, 3, 3] (0 dead)	[128, 64, 3, 3] (0 dead)	[128, 64, 3, 3] (0 dead)
layer2.0.conv2	[128, 128, 3, 3]	[128, 128, 3, 3] (0 dead)	[128, 128, 3, 3] (0 dead)	[128, 128, 3, 3] (0 dead)	[128, 128, 3, 3] (0 dead)
layer2.0.downsample.0	[128, 64, 1, 1]	[128, 64, 1, 1] (0 dead)	[128, 64, 1, 1] (0 dead)	[128, 64, 1, 1] (0 dead)	[128, 64, 1, 1] (0 dead)
layer2.1.conv1	[128, 128, 3, 3]	[128, 128, 3, 3] (0 dead)	[128, 128, 3, 3] (0 dead)	[128, 128, 3, 3] (0 dead)	[128, 128, 3, 3] (0 dead)
layer2.1.conv2	[128, 128, 3, 3]	[128, 128, 3, 3] (0 dead)	[128, 128, 3, 3] (0 dead)	[128, 128, 3, 3] (0 dead)	[128, 128, 3, 3] (0 dead)
layer3.0.conv1	[256, 128, 3, 3]	[191, 128, 3, 3] (65 dead)	[256, 128, 3, 3] (0 dead)	[256, 128, 3, 3] (0 dead)	[256, 128, 3, 3] (0 dead)
layer3.0.conv2	[256, 256, 3, 3]	[164, 191, 3, 3] (92 dead)	[256, 256, 3, 3] (0 dead)	[256, 256, 3, 3] (0 dead)	[256, 256, 3, 3] (0 dead)
layer3.0.downsample.0	[256, 128, 1, 1]	[159, 128, 1, 1] (97 dead)	[256, 128, 1, 1] (0 dead)	[256, 128, 1, 1] (0 dead)	[256, 128, 1, 1] (0 dead)
layer3.1.conv1	[256, 256, 3, 3]	[101, 164, 3, 3] (155 dead)	[256, 256, 3, 3] (0 dead)	[256, 256, 3, 3] (0 dead)	[256, 256, 3, 3] (0 dead)
layer3.1.conv2	[256, 256, 3, 3]	[142, 101, 3, 3] (114 dead)	[256, 256, 3, 3] (0 dead)	[256, 256, 3, 3] (0 dead)	[256, 256, 3, 3] (0 dead)
layer4.0.conv1	[512, 256, 3, 3]	[138, 142, 3, 3] (374 dead)	[512, 256, 3, 3] (0 dead)	[512, 256, 3, 3] (0 dead)	[512, 256, 3, 3] (0 dead)
layer4.0.conv2	[512, 512, 3, 3]	[256, 138, 3, 3] (256 dead)	[512, 512, 3, 3] (0 dead)	[512, 512, 3, 3] (0 dead)	[512, 512, 3, 3] (0 dead)
layer4.0.downsample.0	[512, 256, 1, 1]	[400, 142, 1, 1] (112 dead)	[512, 256, 1, 1] (0 dead)	[512, 256, 1, 1] (0 dead)	[512, 256, 1, 1] (0 dead)
layer4.1.conv1	[512, 512, 3, 3]	[69, 256, 3, 3] (443 dead)	[512, 512, 3, 3] (0 dead)	[505, 512, 3, 3] (7 dead)	[512, 512, 3, 3] (0 dead)
layer4.1.conv2	[512, 512, 3, 3]	[145, 69, 3, 3] (367 dead)	[505, 512, 3, 3] (7 dead)	[495, 505, 3, 3] (17 dead)	[512, 512, 3, 3] (0 dead)
fc	[10, 512]	[10, 145] (0 dead)	[10, 505] (0 dead)	[10, 495] (0 dead)	[10, 512] (0 dead)

G Initialization Sensitivity & Architectural Layer Mapping

G.1 Initialization Sensitivity Ablations

Table 12: Weight initialization scheme ablation evaluation for VGG-16 and ResNet-18 on CIFAR-10.

Model	Initialization Scheme	Pruning Threshold τ	Final Sparsity (%)	Final Test Acc (%)
ResNet-18	Kaiming Normal	$\tau = 5\text{e-}5$	95.41%	75.10%
ResNet-18	Kaiming Uniform	$\tau = 5\text{e-}5$	95.35%	76.08%
ResNet-18	Xavier Normal	$\tau = 5\text{e-}5$	95.66%	77.05%
ResNet-18	Xavier Uniform	$\tau = 5\text{e-}5$	95.54%	76.72%
ResNet-18	Orthogonal	$\tau = 5\text{e-}5$	95.75%	76.51%
ResNet-18	Normal ($\sigma = 0.02$)	$\tau = 5\text{e-}5$	95.96%	77.26%
ResNet-18	Normal ($\sigma = 0.1$)	$\tau = 5\text{e-}5$	94.67%	74.42%
VGG-16	Kaiming Normal	$\tau = 5\text{e-}6$	84.86%	84.80%
VGG-16	Kaiming Uniform	$\tau = 5\text{e-}6$	85.03%	85.30%
VGG-16	Xavier Normal	$\tau = 5\text{e-}6$	86.58%	84.84%
VGG-16	Xavier Uniform	$\tau = 5\text{e-}6$	86.73%	85.38%
VGG-16	Orthogonal	$\tau = 5\text{e-}6$	86.61%	85.63%
VGG-16	Normal ($\sigma = 0.02$)	$\tau = 5\text{e-}6$	**100.00%**	**10.00%**
VGG-16	Normal ($\sigma = 0.1$)	$\tau = 5\text{e-}6$	77.68%	83.44%

G.2 Layer Index to Architectural Name Mapping

Table 13: Sequential layer index mapping $(1, 2, 3, \ldots, L)$ to full parameter layer names for ResNet-18, VGG-16, and MiniBERT.

Layer Index	ResNet-18 (CIFAR-10)	VGG-16 (CIFAR-10)	MiniBERT (SST-2)
1	`conv1`	`features.0 (conv1_1)`	`embeddings.word_embeddings`
2	`layer1.0.conv1`	`features.3 (conv1_2)`	`embeddings.position_embeddings`
3	`layer1.0.conv2`	`features.7 (conv2_1)`	`encoder.layer.0.attention.query`
4	`layer1.1.conv1`	`features.10 (conv2_2)`	`encoder.layer.0.attention.key`
5	`layer1.1.conv2`	`features.14 (conv3_1)`	`encoder.layer.0.attention.value`
6	`layer2.0.conv1`	`features.17 (conv3_2)`	`encoder.layer.0.output.dense`
7	`layer2.0.conv2`	`features.20 (conv3_3)`	`encoder.layer.0.intermediate.dense`
8	`layer2.0.downsample.0`	`features.24 (conv4_1)`	`encoder.layer.1.attention.query`
9	`layer2.1.conv1`	`features.27 (conv4_2)`	`encoder.layer.1.attention.key`
10	`layer2.1.conv2`	`features.30 (conv4_3)`	`encoder.layer.1.attention.value`
11	`layer3.0.conv1`	`features.34 (conv5_1)`	`encoder.layer.1.output.dense`
12	`layer3.0.conv2`	`features.37 (conv5_2)`	`encoder.layer.1.intermediate.dense`
13	`layer3.0.downsample.0`	`features.40 (conv5_3)`	`classifier`
14	`layer3.1.conv1`	`classifier.0 (fc1)`	–
15	`layer3.1.conv2`	`classifier.3 (fc2)`	–
16	`layer4.0.conv1`	`classifier.6 (fc3)`	–
17	`layer4.0.conv2`	–	–
18	`layer4.0.downsample.0`	–	–
19	`layer4.1.conv1`	–	–
20	`layer4.1.conv2`	–	–
21	`fc`	–	–