# PANCAKE: PAIRWISE NEURON CONSOLIDATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Memory and inference-time efficiency are major bottlenecks in scaling neural networks. Structured pruning reduces these costs by removing whole neurons or channels, but most methods formulate pruning as selecting a subset of units to delete, often followed by downstream reconstruction or fine-tuning to recover lost performance. We introduce **PANCake (PAirwise Neuron Consolidation)**, a post-training structured pruning method that instead compresses hidden representations by progressively consolidating redundant neurons. PANCake greedily identifies pairs of neurons with similar activation behavior and merges them, producing a nested hierarchy of representations while avoiding combinatorial subset selection. Across image classification and language modeling, PANCake preserves performance particularly well without global downstream reoptimization. At 20% removal, PANCake retains 45.8% ImageNet top-1 accuracy on ResNet-50, compared with 14.0% when OSSCAR-selected neurons are deleted without downstream optimization and below 1.5% for magnitude pruning. On Pythia-2.8B, it achieves a perplexity of 19.1, compared with 60.3 for OSSCAR-selected deletion. With downstream repair, PANCake remains within 0.5 perplexity of OSSCAR on Pythia-2.8B, Qwen2.5-7B, and Llama-3.1-8B at 20% removal, despite selecting different neurons. These results show that explicitly exploiting redundancy through pairwise consolidation provides an effective alternative to deletion-based structured pruning.

## 1 INTRODUCTION

Modern neural networks achieve remarkable predictive performance, but their increasing scale imposes substantial memory, computational, and energy costs during both training and inference (Vaswani et al., 2019; Han et al., 2015a). At the same time, trained networks often can be compressed substantially: many weights can be removed with little loss in predictive performance (LeCun et al., 1997; Han et al., 2015b), and learned representations have been observed to possess an intrinsic dimension substantially smaller than the width of the corresponding layer (Ansuini et al., 2019). This raises a question: *Which neurons can be removed or consolidated, and how should the remaining parameters be adjusted, while minimizing the resulting perturbation to the hidden representation?*

Beyond its practical role in compression, neuron pruning provides a lens through which to study how each neuron contributes to the hidden representation. Ablation and subset-selection studies suggest that task-relevant information can be distributed across neurons (Morcos et al., 2018; Nanda et al., 2023). In this work, we assume a neuron is redundant not merely when its activation or weights are small, but when its contribution to the hidden representation can be recovered from the remaining neurons.

We define four steps for pruning: *selection, consolidation, stopping, and downstream restructuring*. First, selecting $k$ neurons from a layer of width $H$ entails a search over $\binom{H}{k}$ subsets, making exact selection combinatorial (Yu et al., 2022; Meng et al., 2024). Practical methods therefore rely on approximate criteria, ranging from parameter magnitudes to data-dependent measures of activation (Li et al., 2016; Molchanov et al., 2016; He et al., 2017; Meng et al., 2024). Second, unless we are merging neurons or parameters in a layer, in pruning methods we often do not adjust the parameters in the same layer that we are pruning. Third, many methods assume a prescribed global or layerwise pruning ratio. It is important to know when to stop pruning before we impose substantial damage to the neural network (In this paper, the stopping rule is not our focus). Finally, removing a neuron



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

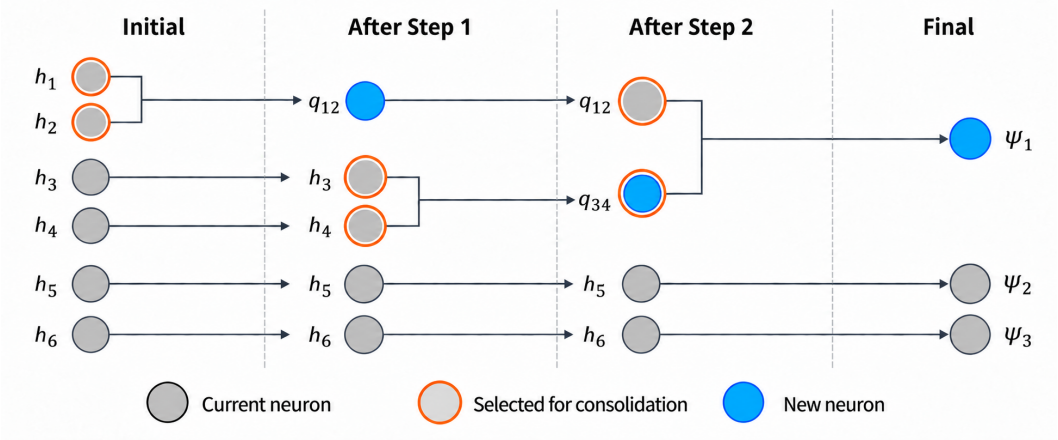


Figure 1: Schematic of PANCake pruning algorithm performing two iterations within a single layer. At each iteration, the algorithm consolidates the two most similar neurons.

perturbs both its representation and downstream computation. This damage may be reduced by updating the surviving parameters through fine-tuning or least-squares reconstruction (Hassibi et al., 1993; He et al., 2017; Kim et al., 2020; Meng et al., 2024).

Existing pruning methods seek to optimize different quantities when removing parameters. Optimal Brain Damage and Optimal Brain Surgeon estimate the increase in the training loss caused by removing individual weights using local second-order approximations (LeCun et al., 1997; Hassibi et al., 1993). Magnitude-based methods instead use parameter norms as proxies for importance (Han et al., 2015b; Li et al., 2016). OSSCAR uses the output of the immediate next layer to optimize pruning Meng et al. (2024). More recently, HOPE represents neurons as Hilbert–Schmidt operators and measures the functional distortion introduced by pruning or merging in an operator space (Mobahi & Bartlett, 2026).

These objectives address different notions of optimizing while pruning: the training loss, parameter magnitude, downstream computation, or the function implemented by a neuron. In contrast, our objective is to minimize the perturbation in the hidden representation. We ask:

*How can we progressively compress a hidden layer while controlling the damage to the hidden representation, when the damage caused by consolidating neurons depends on which other neurons have already been removed/consolidated?*

To address this question, we introduce PANCake, PAirwise Neuron Consolidation, a greedy bottom-up hierarchical consolidation procedure that begins with the original neurons and repeatedly consolidates the pairs that are functionally similar given a sample of dataset (Fig. 1). Each consolidation produces a valid coarser representation, yielding a nested hierarchy that can be cut at any desired width.

Our contributions are as follows:

1. We formulate neuron pruning as hidden representation compression while minimizing the reconstruction error of the original representation from the pruned representation.
2. We introduce PANCake, a greedy bottom-up hierarchical neuron-compression algorithm that constructs a nested hierarchy of representations. At each step, PANCake is locally optimal with respect to the proposed pairwise consolidation cost.
3. Empirically, we evaluate PANCake against established pruning baselines across vision and language models, demonstrating effective neuron removal with minimal downstream restructuring. Its neuron selection on average is at least twice as fast as OSSCAR. With no downstream restructuring, our method removes 20% of the channels of ResNet-50 while retaining 45.8% ImageNet top-1 accuracy, compared to 14.0% when the neurons selected by OSSCAR are deleted and below 1.5% for magnitude pruning. On Pythia-2.8B it removes

20% of the MLP neurons at a perplexity of 19.1, where deleting OSSCAR's selection gives 60.3.

## 2 Related Work

Neural network pruning spans a broad literature, commonly distinguished as unstructured or structured and by whether pruning occurs during training, at initialization, or post-training, uses calibration data, or requires subsequent fine-tuning (Han et al., 2015b; Wen et al., 2016; Blalock et al., 2020; Frankle & Carbin, 2018). Structured methods remove entire filters, channels, neurons, heads, or other computational groups (Li et al., 2016; Liu et al., 2017; He et al., 2018a; Fang et al., 2023), while automated methods optimize layerwise sparsity or architecture under resource constraints (He et al., 2018c; Liu et al., 2019). Rather than organizing this literature by how pruning is performed, we focus on three questions most relevant to our formulation: what quantity is optimized, whether information from removed units is discarded or reassigned, and whether multiple removals can be evaluated independently.

**Minimizing changes in loss or outputs.** Loss-based methods estimate pruning damage using second-order approximations (LeCun et al., 1997; Hassibi et al., 1993), Taylor criteria (Molchanov et al., 2016; 2019), curvature estimates (Singh & Alistarh, 2020), initialization-time criteria (Lee et al., 2018; Wang et al., 2020; Tanaka et al., 2020), or magnitude and learned-sparsity proxies (Han et al., 2015b; Li et al., 2016; Liu et al., 2017; Sanh et al., 2020). Reconstruction-based methods instead preserve downstream responses (Luo et al., 2017; He et al., 2017; Yu et al., 2017). Recent post-training approaches optimize related local reconstruction or loss objectives (Frantar & Alistarh, 2022; 2023; Kurtic et al., 2023; Meng et al., 2024). Unlike downstream least-squares approaches (OSSCAR, ThiNet), PANCake preserves same-layer representations directly in activation space.

**Neuron redundancy and consolidation.** Redundancy-based methods remove units whose contributions can be recovered from others. Prior approaches merge similar neurons (Srinivas & Babu, 2015; Kim et al., 2020), remove geometrically redundant filters (He et al., 2018b), learn removable groups (Chen et al., 2021; Xia et al., 2022), or coordinate coupled removals (Fang et al., 2023; Ma et al., 2023). HOPE formulates pruning and merging through projection in a Hilbert space (Mobahi & Bartlett, 2026). We characterize redundancy directly in activation space, conditional on the representation retained after consolidation.

A comprehensive comparison of the neural network pruning literature is provided in Appendix A.

## 3 Problem Formulation and Theoretical Framework

**Layer representation.** For any layer, let $X$ be the random variable representing the output of the preceding layer, with distribution $P$. Within each layer, the post-activation response of neuron $i \in [d]$ is $h_i(X) = \sigma(w_i X + b_i)$, and the complete layer representation is $\mathbf{h}(X) = (h_1(X), \ldots, h_d(X))^\top \in \mathbb{R}^d$. We assign each neuron an importance weight, $a_i > 0$ which will serve as a proxy for downstream impact of the neuron (i.e. if a neuron has high importance weight, it's outgoing weights to the next layer are high). Importance weights depend on model architecture and will be defined explicitly in Section 4.

A width-$k$ compressed representation of a layer consists of scalar responses $\psi_1(X), \ldots, \psi_k(X)$. Depending on the consolidation rule, these responses may be the outputs of original neurons or combinations of them.

**Selecting the optimal subset is NP-hard.** Suppose $K \subset [d]$ is the subset of retained neurons and thus $\bar{K} = [d] \setminus K$ is the subset of removed neurons. Let $\mathbf{h}_K$ be a layer representation including only the set of neurons $K$, and let $\mathbf{h}_{\bar{K}}$ be a layer representation including the set of neurons $[d] \setminus K$. We define the reconstruction error associated with $K$ as

$$\mathcal{E}(K) := \min_{B \in \mathbb{R}^{|\bar{K}| \times |K|}} \mathbb{E}_P \left[ \left\| A_{\bar{K}}^{1/2} \left( \mathbf{h}_{\bar{K}} - B\mathbf{h}_K \right) \right\|_F^2 \right], \qquad \bar{K} = [d] \setminus K, \tag{1}$$

where $A_{\bar{K}} = \mathrm{diag}\big((a_r)_{r\in\bar{K}}\big) \in \mathbb{R}^{|\bar{K}|\times|\bar{K}|}$. While this looks like the isolated error of a single layer, the importance weights $a_r$ of the removed neurons scale the loss terms by how much impact those removed neurons had in following layers.

Selecting the optimal subset, $K^\star \in \arg\min_{K\subset[d],|K|=k} \mathcal{E}(K)$, is an NP-hard combinatorial optimization problem. In particular, under the empirical distribution and with uniform importance weights, this objective reduces to the standard column subset selection problem, which is NP-hard (Shitov, 2017). We therefore replace exact subset optimization with greedy pairwise consolidations.

**Pairwise consolidation.** Consider consolidating neurons $h_i$ and $h_j$ into a single neuron $q_{ij}$. To simplify notation, we define $\|f\|_P^2 = \mathbb{E}_{X\sim P}[f(X)^2]$ for any function $f$. The reconstruction error induced by this consolidation is

$$\mathcal{E}_{\mathrm{pair}}(i,j;q_{ij}) := a_i\|h_i - q_{ij}\|_P^2 + a_j\|h_j - q_{ij}\|_P^2. \tag{2}$$

We define the consolidated pair importance weight, importance weighted average neuron, and empirical weighted pair distance as

$$a_{ij} = a_i + a_j, \qquad \mu_{ij} = \frac{a_i h_i + a_j h_j}{a_{ij}}, \qquad \Delta(i,j) = \frac{a_i a_j}{a_{ij}}\|h_i - h_j\|_P^2. \tag{3}$$

It can be shown that we can express (2) as

$$\mathcal{E}_{\mathrm{pair}}(i,j;q_{ij}) = \Delta(i,j) + a_{ij}\|q_{ij} - \mu_{ij}\|_P^2. \tag{4}$$

See Appendix C for the derivation. We write it in this way because in practice, when we select a pair for consolidation, we do not optimize for $q_{ij}$, only the selection of $i, j$. Therefore separating the error like this allows us to minimize $\Delta$ independently.

Once $i$ and $j$ are selected, we choose $q_{ij}$ to be either the weighted average neuron $\mu_{ij}$ or the neuron with greater importance weight. The former makes the second term in equation 4 vanish. For the latter, suppose without loss of generality that $a_i \geq a_j$ and choose $q_{ij} = h_i$. Then

$$a_{ij}\|h_i - \mu_{ij}\|_P^2 = \frac{a_j}{a_i}\Delta(i,j) \leq \Delta(i,j). \tag{5}$$

Thus, retaining the neuron with greater importance weight adds at most $\Delta(i,j)$ beyond the weighted-average consolidation error, and consequently

$$\mathcal{E}_{\mathrm{pair}}(i,j;h_i) \leq 2\Delta(i,j). \tag{6}$$

## 4 Algorithm Design and Implementation

We call our method PANCake (PAirwise-Neuron-Consolidation-*ake*). For each layer, working from the first to the last, PANCake iteratively consolidates the most similar pair of neurons into a new neuron until the target layer width is reached. Once the consolidation is finished, it restructures the outgoing weights in a way that will be discussed in Subsection 4.3.

### 4.1 Importance Weights

For each neuron $i$, its importance weight, $a_i > 0$, determines how important that neuron is for the input to the next layer. For a ReLU network, we define the importance weight as

$$a_i = ||w_i^{out}||_2||w_i^{in}||_2 \tag{7}$$

For networks with GELU and other non-homogeneous activation functions, we define the importance weight as

$$a_i = \mathbb{E}_{X\sim P}\left[h_i(X)^2\right]^{1/2}||w_i^{out}||_2 \tag{8}$$

The reason for the separation of homogeneous and non-homogeneous activation functions is that ReLU is positively homogeneous ($\sigma(\alpha x) = \alpha\sigma(x)$ for $\alpha \geq 0$), meaning the magnitude of the activation scales directly with the norm of the incoming weights $\|w_i^{\text{in}}\|_2$, making $\|w_i^{\text{in}}\|_2\|w_i^{\text{out}}\|_2$ a scale-invariant proxy for neuron influence under positive rescaling. In contrast, non-homogeneous activations like GELU do not scale linearly with input magnitude ($\sigma(\alpha x) \neq \alpha\sigma(x)$), causing input weight norms to misrepresent actual activation scale. Consequently, non-homogeneous networks require explicitly measuring the data-dependent activation norm $\mathbb{E}_{X\sim P}\left[h_i(X)^2\right]^{1/2}$.

### 4.2 GREEDY PAIRWISE CONSOLIDATION

As described in Section 3, our goal at each iteration is to select a pair of neurons that minimize the empirical weighted pair distance, $\Delta$. $(i^*, j^*) = \arg\min_{i,j\in[d],i\neq j} \Delta(i,j)$ can be computed easily for each pair in $\mathcal{O}(d^2)$. Once $(i^*, j^*)$ is found, it remains to compose the new neuron $q_{ij}$ (we have dropped the $^*$ for simplicity. We can choose $q_{ij}$ as the importance weighted average neuron or simply select either $h_i$ or $h_j$:

$$q_{ij} = \begin{cases} \dfrac{a_i h_i + a_j h_j}{a_i + a_j}, & \text{if using the weighted average} \\ h_i \quad (\text{or } h_j), & \text{if selecting directly (medoid)} \end{cases} \tag{9}$$

The choice largely depends on the architecture, and both are examined in Appendix B.

Once the new neuron has been constructed, we simply delete one of the original neurons, either $h_i$ or $h_j$, and set $d = d - 1$. Once $d = k$, we continue to the output weight reconstruction.

### 4.3 OUTGOING WEIGHT RECONSTRUCTION

Since our layer size has reduced from $d$ to $k$, we must restructure the outgoing weights so that the neurons can interact with the next layer correctly. We have implemented two separate methods described as follows.

**Direct Weight Addition** Let $W \in \mathbb{R}^{m\times d}$ denote the original outgoing weight matrix, where $w_l \in \mathbb{R}^m$ represents the outgoing weight vector for original neuron $h_l$. Over successive iterations, a consolidated neuron $q$ may incorporate multiple original neurons. Let $\mathcal{C}(q) \subseteq \{1, \ldots, d\}$ denote the set of original neuron indices that have been merged into $q$. The outgoing weight $w_q \in \mathbb{R}^m$ for neuron $q$ is defined as the sum of all original constituent weights:

$$w_q = \sum_{l\in\mathcal{C}(q)} w_l \tag{10}$$

Equivalently, this can be computed dynamically during each step: whenever pair $(i, j)$ merges into $q_{ij}$, its updated weight is assigned as $w_{q_{ij}} = w_i + w_j$. This method requires no calibration data and adds negligible computational overhead ($\mathcal{O}(m)$ per merge), but it operates on the assumption that representations remain approximately linear over multiple consolidations.

**OSSCAR Reconstruction** The second approach adapts the refitting strategy from OSSCAR (Meng et al., 2024), performing a global least-squares optimization on the outgoing weights once all pairwise consolidations are complete. Let $H \in \mathbb{R}^{N\times d}$ be the initial activation matrix across $N$ calibration samples, producing the baseline output $Y = HW^\top \in \mathbb{R}^{N\times m}$. After $d - k$ consolidation iterations, the remaining $k$ active neurons form the reduced activation matrix $\hat{H} \in \mathbb{R}^{N\times k}$. The re-optimized weight matrix $W_{\text{rec}} \in \mathbb{R}^{m\times k}$ is obtained by solving:

$$W_{\text{rec}} = \arg\min_{W'} \frac{1}{2N} \|Y - \hat{H}W'^\top\|_F^2 \tag{11}$$

which has the closed-form solution $W_{\text{rec}}^\top = (\hat{H}^\top\hat{H})^{-1}\hat{H}^\top Y$. Because OSSCAR refits the output weights jointly at the end of the reduction process, it inherently accounts for neurons that have undergone multiple consolidations without accumulating error step-by-step. However, this comes at the cost of higher computational complexity ($\mathcal{O}(Nk^2 + k^3)$ for matrix inversion/solving) and requires access to a calibration dataset.

The full algorithm is described in Algorithm 1.

## 5 EXPERIMENTS

Classification models are evaluated by top-1 accuracy, whereas language models are evaluated by WikiText-2 perplexity. Perplexity is shown on a logarithmic scale. Curves report the mean across the seeds indicated in each panel, and shaded regions span the minimum to maximum across seeds at each removal fraction. In the overlap experiment, the band instead spans the minimum to maximum across layers for seed 0.

**Algorithm 1** Layer-by-Layer PANCake Framework

**Require:** Network layers $l = 1 \dots L$, initial widths $d_l$, target widths $k_l$, dataset $P$, mode $\in$ $\{\text{Direct}, \text{OSSCAR}\}$

**for** $l \leftarrow 1$ **to** $L$ **do**
  Compute importance weights $a_i$ for all $i \in \{1, \dots, d_l\}$ using equation 7 or equation 8
  Initialize active set $\mathcal{S} \leftarrow \{1, \dots, d_l\}$
  **while** $|\mathcal{S}| > k_l$ **do**
    $(i^*, j^*) \leftarrow \arg\min_{i,j \in \mathcal{S}, i \neq j} \Delta(i, j)$
    Merge pair $(i^*, j^*)$ into $q$: compute $h_q$ and set $a_q \leftarrow a_{i^*} + a_{j^*}$
    **if** mode $=$ Direct **then**
      $w_q \leftarrow w_{i^*} + w_{j^*}$
    **end if**
    $\mathcal{S} \leftarrow (\mathcal{S} \setminus \{i^*, j^*\}) \cup \{q\}$
  **end while**
  Form reduced activation matrix $\hat{H}^{(l)}$ from active neurons in $\mathcal{S}$
  **if** mode $=$ OSSCAR **then**
    $W_{\text{rec}}^{(l)\top} \leftarrow ((\hat{H}^{(l)})^\top \hat{H}^{(l)})^{-1} (\hat{H}^{(l)})^\top Y^{(l)}$
  **else**
    $W_{\text{rec}}^{(l)} \leftarrow [w_q]_{q \in \mathcal{S}}$
  **end if**
  Forward-propagate reduced activations $\hat{H}^{(l)}$ to layer $l + 1$
**end for**
**return** Reconstructed layer weights $\{W_{\text{rec}}^{(l)}\}_{l=1}^{L}$

### 5.1 Neuron Removal Without Downstream Reoptimization

We first compare pruning strategies without globally reoptimizing the weights of the immediately following layer. The baselines are random deletion, magnitude pruning, and OSSCAR (Meng et al., 2024). For neuron $j$, magnitude pruning assigns the score $s_j^{\text{mag}} = \left\|w_j^{\text{in}}\right\|_2 \left\|w_j^{\text{out}}\right\|_2$, and removes neurons with the smallest scores. For convolutional channels, we use the Frobenius norms of the corresponding incoming and outgoing weight groups.

OSSCAR selects neurons by minimizing reconstruction error at the output of the immediately following layer while reoptimizing its surviving weights. To isolate the effect of neuron selection, we retain the removal set produced by OSSCAR but discard its reoptimized weights. Thus, no method in this experiment is evaluated with globally reoptimized downstream weights.

PANCake, after selecting a similar pair, applies either the merge or medoid consolidation rule and transfers the absorbed neuron's outgoing weights to the retained neuron using the sum rule. This local adjustment requires no least-squares fitting. Each panel reports the better-performing consolidation rule, as indicated in the panel.

PANCake performs best for both ResNets and Pythia models (Fig. 2). OSSCAR removal performs best on ViT-B/16 and OPT-6.7B, whereas magnitude pruning is strongest on Qwen2.5-7B and Llama-3.1-8B. On ViT-B/16, random deletion is competitive with PANCake and magnitude pruning at moderate removal fractions but deteriorates sharply at the largest fraction. These results show that same-layer representational similarity provides a highly informative pruning criterion for several architectures, although it is not uniformly the strongest criterion across all models.

To determine whether the methods succeed or fail on the same neurons, we also measure the overlap between their removal sets. For two methods that remove sets $S_1$ and $S_2$ of equal size, we define $\text{Overlap}(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1|}$.

Despite their different objectives, PANCake and OSSCAR select substantially overlapping neurons, with overlap above chance across all evaluated architectures (Table 1). PANCake and magnitude pruning also overlap above chance for most models, although their agreement is near or below chance on ViT-B/16 and approaches chance at moderate removal fractions on OPT-6.7B. Thus, PANCake, OSSCAR, and weight magnitude capture partially shared structure, but their non-overlapping selections remain consequential for final performance. Because chance overlap increases mechani-

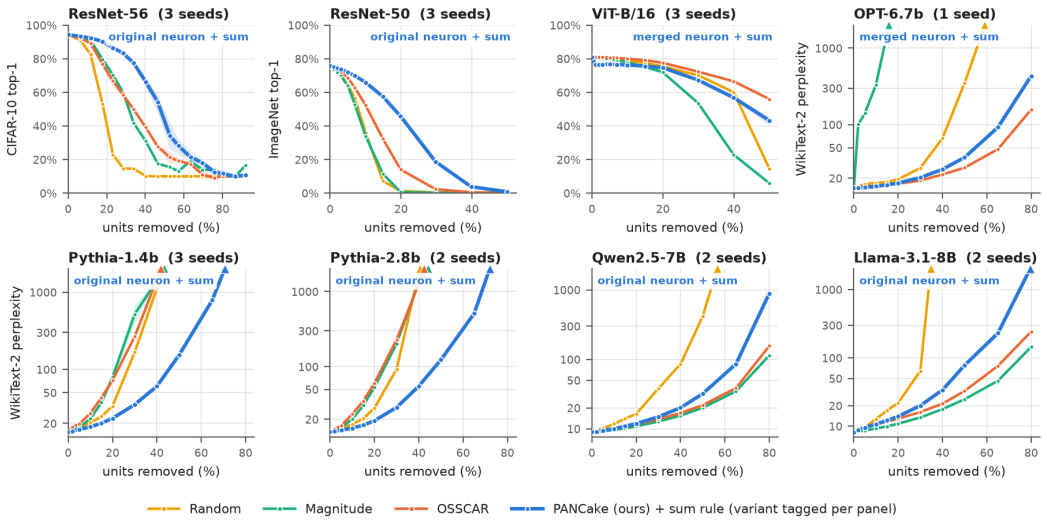


Figure 2: Neuron removal without global least-squares repair. Random, magnitude, and OSSCAR-selected units are deleted without reoptimizing the surviving downstream weights; PANCake applies only its local sum rule. Higher accuracy and lower perplexity are better. Bands show the minimum-to-maximum range across seeds.

cally with the removal fraction, the low-to-moderate removal regime is the most informative part of this comparison.

Table 1: Removed-set overlap $|A \cap B| / \min(|A|, |B|)$ between PANCake's removed units and OSSCAR's (resp. magnitude's) at the same width, mean over layers, one seed, at two removal fractions per model. Chance is the overlap of two random sets of that size.

| | ResNet-56 | | ResNet-50 | | ViT-B/16 | | OPT-6.7b | | Pythia-1.4b | | Pythia-2.8b | | Qwen2.5-7B | | Llama-3.1-8B | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| removed | 18% | 42% | 20% | 40% | 20% | 40% | 20% | 40% | 20% | 40% | 20% | 40% | 20% | 40% | 20% | 40% |
| PANCake vs OSSCAR | 0.61 | 0.69 | 0.46 | 0.55 | 0.70 | 0.68 | 0.82 | 0.82 | 0.36 | 0.48 | 0.36 | 0.48 | 0.56 | 0.62 | 0.49 | 0.53 |
| PANCake vs magnitude | 0.47 | 0.62 | 0.26 | 0.51 | 0.13 | 0.33 | 0.26 | 0.36 | 0.40 | 0.50 | 0.40 | 0.48 | 0.59 | 0.62 | 0.57 | 0.55 |
| chance | 0.18 | 0.42 | 0.20 | 0.40 | 0.20 | 0.40 | 0.20 | 0.40 | 0.20 | 0.40 | 0.20 | 0.40 | 0.20 | 0.40 | 0.20 | 0.40 |

### 5.2 Neuron removal with downstream repair

We next evaluate the effect of OSSCAR next-layer optimization. All pruning methods use empirical least-squares reconstruction with a relative ridge or damping coefficient of $0.01$. Random and magnitude selection use zero-centered ridge repair. For PANCake, the ridge penalty is instead centered on the weights obtained from the sum rule, using the same $\lambda$; this convention is used for both the merge and medoid consolidation rules. OSSCAR retains its default damped reconstruction. Therefore, the methods share the same relative regularization scale but retain their method-specific repair procedures.

PANCake gains the least increase in the performance for ResNet architecture and Pythia models. OSSCAR gains a performance boost from its optimization for ResNet and Pythia. This is consistent with its downstream objective, which directly selects neurons to minimize consumer-layer reconstruction error. Qwen2.5-7B and Llama-3.1-8B are the exceptions, for which magnitude selection followed by ridge repair are enough for pruning. Downstream optimization also substantially improves random selection at low-to-moderate removal fractions, particularly for the language models, although the selection criterion becomes increasingly important under aggressive compression.

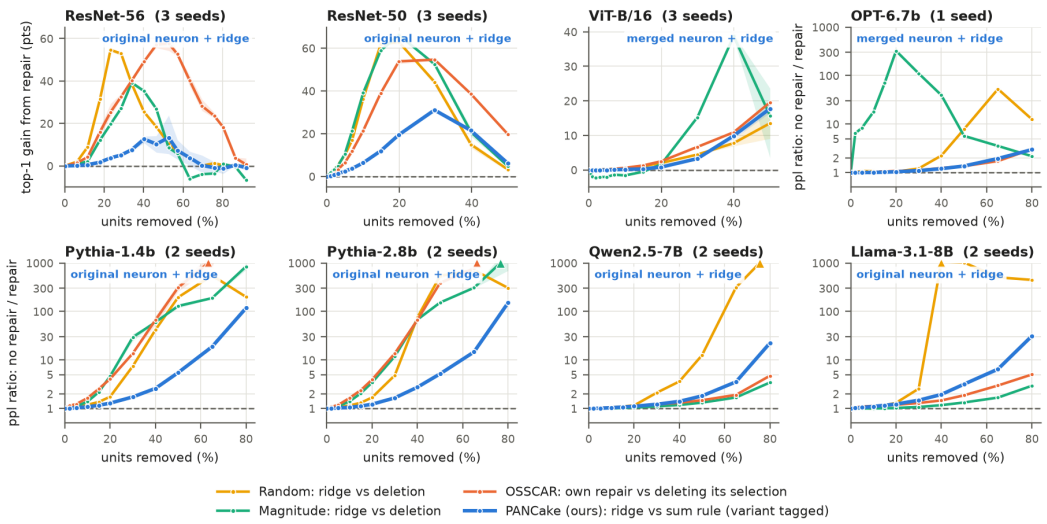


Figure 3: Performance improvement from OSSCAR-like next-layer optimization. For classification models, we report the change in top-1 accuracy; for language models, we report the ratio of perplexity before and after repair.

### 5.3 Ablation: Removing Both Members of a Similar Pair

PANCake assumes that one member of a similar pair can represent the pair after the absorbed neuron's outgoing weights are transferred to the survivor. We test this mechanism using four interventions at matched removal fractions: standard PANCake with the sum rule; deletion of the absorbed member without modifying the survivor; deletion of both members of each PANCake-selected pair; and random deletion.

The direct test is whether deleting both members is more damaging than retaining and updating the survivor. This prediction holds for ResNets, ViT-B/16, OPT-6.7B, Pythia models, and Llama-3.1-8B, with particularly large differences for Pythia and Llama. Qwen2.5-7B is the main exception, where deleting both members remains competitive with standard PANCake.

The comparison with random deletion is architecture-dependent. Pair deletion is more damaging than random deletion for Pythia and Llama, but less damaging for the ResNets, ViT, OPT, and Qwen. This does not directly contradict the PANCake hypothesis: two similar neurons may be jointly less important than two randomly selected neurons even when retaining one member remains preferable.

### 5.4 Timing Comparison

Table 2 reports the cost of producing the full width curve. PANCake is slower than magnitude pruning because it constructs an activation-space hierarchy, but this hierarchy is computed only once and reused across all target widths. With the sum rule, PANCake remains consistently faster than OSSCAR; when ridge repair is applied, the repeated least-squares solves dominate the runtime and make PANCake slower than OSSCAR on the language models.

## 6 Conclusion

We introduced PANCake, a post-training structured pruning method that views compression as the progressive consolidation of redundant neurons rather than independent neuron deletion. PANCake greedily merges similar neurons based on the defined distance metrics, producing a nested hierarchy of representations while avoiding combinatorial subset selection. Across vision and language models, PANCake preserves performance particularly well without global downstream reoptimization, demonstrating that same-layer representational similarity can provide an informative criterion

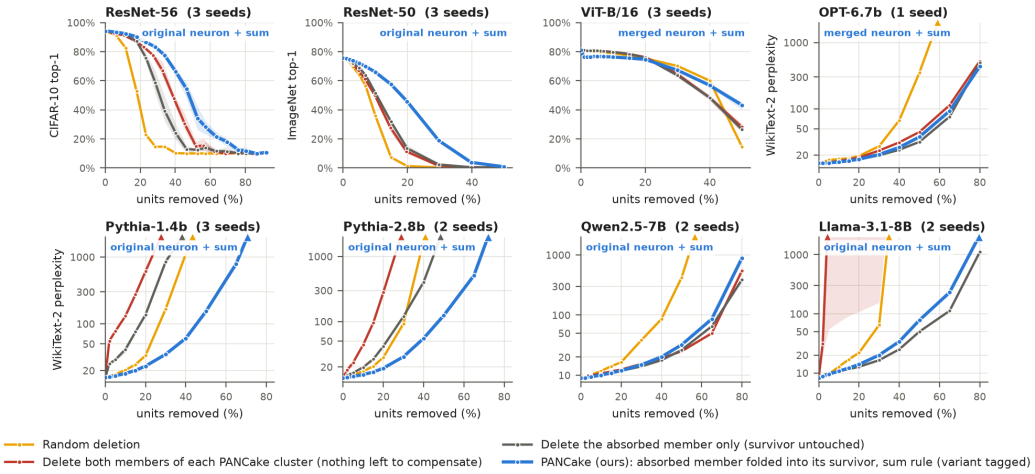


Figure 4: Ablation of PANCake pairwise consolidation. Standard PANCake removes the absorbed member and transfers its outgoing weights to the survivor; the alternative interventions omit this transfer, remove both pair members, or delete units randomly. Higher accuracy and lower perplexity are better. Bands show the minimum-to-maximum range across seeds.

Table 2: Wall-clock cost of producing the full width curve of the figures (10 target widths for the language models, 11 for ImageNet, 16 for CIFAR-10). PANCake builds one dendrogram per model and cuts it at each width, either with the sum rule or followed by the ridge repair; magnitude and OSSCAR select (and repair) anew at every width. Medians over seeds; evaluation excluded; the GPU row names the card each model was measured on.

| | ResNet-56 | ResNet-50 | OPT-6.7b | Qwen2.5-7B | Llama-3.1-8B |
|---|---|---|---|---|---|
| GPU | A100 | A100 | H100 | H100 | H100 |
| Magnitude | 16 s | 12 s | 7 min* | 23 min | 24 min |
| Magnitude + ridge | 45 s | 99 s | 3.1 h | 3.5 h | 2.3 h |
| PANCake + sum rule | 29 s | 56 s | 1.5 h | 1.1 h | 54 min |
| PANCake + ridge | 58 s | 3 min | 6.2 h | 5.8 h | 4.2 h |
| OSSCAR | 6 min | 48 min | 2.5 h | 3.7 h | 2.3 h |

*OPT-6.7b Magnitude: measured on A100.

for identifying redundant neurons. When downstream least-squares repair is applied, PANCake remains competitive with OSSCAR despite selecting a different set of neurons. Together, these results suggest that explicitly exploiting redundancy through neuron consolidation provides a useful framework for structured compression and for studying how information is distributed across hidden representations.

## AI USE STATEMENT

In this work, we used generative AI tools to assist with algorithm implementation, literature search, code generation and debugging, figure and table generation, and language editing and refinement of the manuscript. Generative AI was also used to help improve the clarity, organization, and presentation of text written by the authors. The research questions, methodological ideas, theoretical results, experimental design, interpretation of results, and scientific conclusions were developed and determined by the authors. All AI-assisted code, text, and literature suggestions were reviewed and verified by the authors. In particular, generated code and plotting scripts were manually tested for correctness, and literature suggestions were checked against the original sources. The authors take full responsibility for the final content of the manuscript, including all claims, results, and artifacts produced with the assistance of generative AI.

## REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our theoretical and empirical results, all algorithmic procedures, mathematical proofs, and experimental configurations are fully detailed throughout the manuscript and supplementary material. Specifically, the complete step-by-step logic for the PANCake framework is formalized in Algorithm 1. Complete theoretical proofs and bounds for representation preservation are provided in Appendix C. All empirical evaluations rely on publicly accessible datasets (CIFAR-10, ImageNet, and WikiText-2) and standard open-source pre-trained model architectures (ResNet, ViT, OPT, Pythia, Qwen2.5, and Llama-3.1). Detailed experimental settings—including the $0.01$ relative ridge damping coefficient for output weight reconstruction, seed counts, and GPU hardware environments—are specified in Section 5 and Table 2. Additionally, complete anonymized source code and scripts required to compute activation similarity matrices, build bottom-up hierarchies, and evaluate model performance across pruning fractions are included in the supplementary material.

## REFERENCES


Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 10865–10873, 2024.

Alessio Ansuini, Alessandro Laio, Jakob H. Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. In *Neural Information Processing Systems*, volume 32, 2019.

Davis W. Blalock, J. J. Ortiz, Jonathan Frankle, and J. Guttag. What is the state of neural network pruning? In *Conference on Machine Learning and Systems*, volume 2, 2020.

Tianyi Chen, Bo Ji, Tianyu Ding, Biyi Fang, Guanyi Wang, Zhihui Zhu, Luming Liang, Yixin Shi, Sheng Yi, and Xiao Tu. Only train once: A one-shot neural network training and pruning framework. In *Neural Information Processing Systems*, volume 34, 2021.

Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. DepGraph: Towards any structural pruning. In *Computer Vision and Pattern Recognition*, pp. 16091–16101. IEEE, 2023. doi: 10.1109/CVPR52729.2023.01544.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In *Advances in Neural Information Processing Systems 35*, volume 35, pp. 4475–4488. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2022. doi: 10.52202/068431-0323.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10323–10337. PMLR, 2023. doi: 10.48550/arXiv.2301.00774.

Marwa El Halabi, Suraj Srinivas, and Simon Lacoste-Julien. Data-efficient structured pruning via submodular optimization. In *Advances in Neural Information Processing Systems 35*, volume 35, pp. 36613–36626. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2022. doi: 10.52202/068431-2653.

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*, 2015a.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network. In *Neural Information Processing Systems*, volume 28, 2015b.

Babak Hassibi, David G. Stork, and Gregory Wolff. Optimal brain surgeon: Extensions and performance comparisons. In *Neural Information Processing Systems*, volume 6, 1993.

Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *International Joint Conference on Artificial Intelligence*, pp. 2234–2240. International Joint Conferences on Artificial Intelligence Organization, 2018a. doi: 10.24963/ijcai.2018/309.

Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Computer Vision and Pattern Recognition*, pp. 4335–4344. IEEE, 2018b. doi: 10.1109/CVPR.2019.00447.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *IEEE International Conference on Computer Vision*, pp. 1398–1406. IEEE, 2017. doi: 10.1109/ICCV.2017.155.

Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. AMC: AutoML for model compression and acceleration on mobile devices. In *European Conference on Computer Vision*, pp. 815–832. Springer International Publishing, 2018c. doi: 10.1007/978-3-030-01234-2_48.

Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *International Conference on Machine Learning*, 2016.

Woojeong Kim, Suhyun Kim, Mincheol Park, and Geonseok Jeon. Neuron merging: Compensating for pruned neurons. In *arXiv (Cornell University)*, volume 33, pp. 585–595. Cornell University, 2020. doi: 10.48550/arxiv.2010.13160.

Eldar Kurtic, Elias Frantar, and Dan Alistarh. ZipLM: Inference-aware structured pruning of language models. In *Neural Information Processing Systems*, volume 36, pp. 65597–65617. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2023. doi: 10.52202/075280-2863.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Journal of Japan Society for Fuzzy Theory and Systems*, volume 9, pp. 50. Japan Society for Fuzzy Theory and Intelligent Informatics, 1997. doi: 10.3156/jfuzzy.9.1_50_2.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. SNIP: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2018.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2016.

Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. HRank: Filter pruning using high-rank feature map. In *Computer Vision and Pattern Recognition*, pp. 1526–1535. IEEE, 2020. doi: 10.1109/cvpr42600.2020.00160.

Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. MetaPruning: Meta learning for automatic neural network channel pruning. In *IEEE International Conference on Computer Vision*, pp. 3295–3304. IEEE, 2019. doi: 10.1109/ICCV.2019.00339.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *IEEE International Conference on Computer Vision*, pp. 2755–2763. Ieee, IEEE, 2017. doi: 10.1109/ICCV.2017.298.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. ThiNet: A filter level pruning method for deep neural network compression. In *IEEE International Conference on Computer Vision*, pp. 5068–5076. IEEE, 2017. doi: 10.1109/ICCV.2017.541.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-Pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems 36*, volume 36, pp. 21702–21720. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2023. doi: 10.52202/075280-0950.

Xiang Meng, Shibal Ibrahim, Kayhan Behdin, Hussein Hazimeh, Natalia Ponomareva, and Rahul Mazumder. OSSCAR: One-shot structured pruning in vision and language models with combinatorial optimization. In *International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35354–35377. PMLR, 2024. doi: 10.48550/arXiv.2403.12983.

Hossein Mobahi and Peter L. Bartlett. Hilbert operator for progressive encoding (HOPE): A mathematical framework for deconstructing learned representations in deep networks. *Computer Vision and Pattern Recognition*, 2026. doi: 10.48550/arXiv.2607.21366.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *International Conference on Learning Representations*, 2016.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Computer Vision and Pattern Recognition*, pp. 11256–11264. IEEE, 2019. doi: 10.1109/CVPR.2019.01152.

Ari S. Morcos, David G. T. Barrett, Neil C. Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. In *International Conference on Learning Representations*, 2018.

Vedant Nanda, Till Speicher, John Dickerson, Krishna Gummadi, Soheil Feizi, and Adrian Weller. Diffused redundancy in pre-trained representations. In *Advances in Neural Information Processing Systems 36*, volume 36, pp. 4055–4079. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2023. doi: 10.52202/075280-0177.

Siyuan Pan, Linna Zhang, Jie Zhang, Xiaoshuang Li, Liang Hou, and Xiaobing Tu. Layer-adaptive structured pruning guided by latency. *arXiv preprint arXiv:2305.14403*, 2023.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. Movement pruning: Adaptive sparsity by fine-tuning. In *Neural Information Processing Systems*, volume 33, 2020.

Yaroslav Shitov. Column subset selection is np-complete. *Linear Algebra and its Applications*, 610: 52–58, 2017. doi: 10.1016/J.LAA.2020.09.015.

Sidak Pal Singh and Dan Alistarh. WoodFisher: Efficient second-order approximation for neural network compression. In *Neural Information Processing Systems*, volume 33, 2020.

Suraj Srinivas and R. Venkatesh Babu. Data-free parameter pruning for deep neural networks. In *British Machine Vision Conference*, pp. 31.1–31.12. British Machine Vision Association, 2015. doi: 10.5244/C.29.31.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, 2024. doi: 10.48550/arXiv.2306.11695.

Hidenori Tanaka, Daniel Kunin, Daniel L. K. Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Neural Information Processing Systems*, volume 33, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N.Gomez, Lukasz Kaiser, and Illia Polosukhin. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3645–3650. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1355.

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

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Neural Information Processing Systems*, volume 29, 2016.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. Structured pruning learns compact and accurate models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1513–1528. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-long.107.

Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I. Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S. Davis. NISP: Pruning networks using neuron importance score propagation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9194–9203. IEEE, 2017. doi: 10.1109/CVPR.2018.00958.

Xin Yu, Thiago Serra, Srikumar Ramalingam, and Shandian Zhe. The combinatorial brain surgeon: Pruning weights that cancel one another in neural networks. In *International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 2022. doi: 10.48550/arXiv.2203.04466.

Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Neural Information Processing Systems*, volume 31, 2018.

# A TAXONOMY OF PRUNING AND NEURON-MERGING METHODS

This section provides a taxonomy of neural network pruning and neuron-merging methods. We organize prior work according to the structure being removed, the stage at which compression is performed, the quantity that the method seeks to preserve or optimize, and whether pruning requires data or subsequent fine-tuning. We additionally distinguish how methods respond to removal, ranging from direct deletion to reconstruction, compensation, or merging, and whether selection decisions are made independently using fixed scores or depend on other pruning decisions through adaptive, greedy, or joint optimization.

Table 3: Representative taxonomy of neural network pruning and neuron-merging methods. "Data-free" refers specifically to whether examples are required to determine which components to remove or merge, not whether data were used to train the original model. S/U denote structured/unstructured pruning; PT denotes post-training.

| Paper / Method | Year | Type | Pruned object | Stage | Preserved / optimized quantity | Fine-tune? | Data-free selection? | Action | Adjustment after removal | Selection coupling |
|---|---|---|---|---|---|---|---|---|---|---|
| **Classical loss- and weight-based pruning** | | | | | | | | | | |
| Optimal Brain Damage (OBD) (LeCun et al., 1997) | 1990 | U | Weight | PT | Task loss; diagonal second-order approximation | Optional | No (loss/Hessian) | Prune | No analytical compensation | Fixed score |
| Optimal Brain Surgeon (OBS) (Hassibi et al., 1993) | 1993 | U | Weight | PT | Task loss; full second-order approximation | No | No (loss/Hessian) | Prune | Analytical update of surviving weights | Adaptive / sequential |
| Learning Both Weights and Connections (Han et al., 2015b) | 2015 | U | Weight | PT | Magnitude as importance proxy | Yes | Yes (weights only) | Prune | Retrain surviving weights | Fixed ranking |
| Structured Sparsity Learning (Wen et al., 2016) | 2016 | S | Filters / channels / shapes | Train | Task loss + structured sparsity regularization | Joint | No (training data) | Prune | Parameters optimized jointly during training | Joint optimization |
| The Lottery Ticket Hypothesis (Frankle & Carbin, 2018) | 2019 | U | Weights / connections | Train–rewind | Trainability of sparse subnetworks | Retrain | Yes$^{\dagger}$ | Prune | Rewind and retrain surviving subnetwork | Iterative magnitude |
| SNIP (Lee et al., 2018) | 2019 | U | Weight | Init | Task-loss sensitivity at initialization | No$^{*}$ | No (data/gradients) | Prune | None; sparse network subsequently trained | Fixed score |
| GraSP (Wang et al., 2020) | 2020 | U | Weight | Init | Gradient-flow preservation | No$^{*}$ | No (data/gradients) | Prune | None; sparse network subsequently trained | Fixed score |
| SynFlow (Tanaka et al., 2020) | 2020 | U | Weight | Init | Synaptic-flow preservation | No$^{*}$ | Yes | Prune | None; sparse network subsequently trained | Iterative re-scoring |
| WoodFisher (Singh & Alistarh, 2020) | 2020 | U | Weight | PT | Approximate task-loss increase | Optional | No (empirical Fisher) | Prune | OBS-style compensation of surviving weights | Adaptive / sequential |
| Movement Pruning (Sanh et al., 2020) | 2020 | U | Weight | Fine-tune | Task loss / movement of weights during fine-tuning | Joint | No (task data) | Prune | Weights and mask scores learned during fine-tuning | Learned score |
| LAMP (Pan et al., 2023) | 2021 | U | Weight | PT | Approx. model-level $\ell_2$ distortion | Optional | Yes (weights only) | Prune | None unless followed by retraining | Fixed global ranking |
| **Structured neuron, channel, and filter pruning** | | | | | | | | | | |
| Data-Free Parameter Pruning for Deep Neural Networks (Srinivas & Babu, 2015) | 2015 | S | Neuron | PT | Neuron similarity / functional redundancy | No | Yes | Merge-like prune | Outgoing weights modified to absorb removed neuron | Pairwise similarity |
| Network Trimming (Hu et al., 2016) | 2016 | S | Neuron | PT | Activation-frequency / inactivity proxy | Yes | No (activations) | Prune | Fine-tune surviving network | Fixed score |
| Pruning Filters for Efficient ConvNets (Li et al., 2016) | 2017 | S | Filter | PT | Filter-norm importance proxy | Yes | Yes (weights only) | Prune | Fine-tune surviving network | Fixed ranking |

*Continued on next page*

*Table 3 continued*

| Paper / Method | Year | Type | Pruned object | Stage | Preserved / optimized quantity | Fine-tune? | Data-free selection? | Action | Adjustment after removal | Selection coupling |
|---|---|---|---|---|---|---|---|---|---|---|
| ThiNet (Luo et al., 2017) | 2017 | S | Channel | PT | **Downstream-layer output reconstruction** | Yes | No (activations) | Prune | Least-squares reconstruction + fine-tuning | Greedy set-dependent |
| Channel Pruning for Accelerating Very Deep Networks (He et al., 2017) | 2017 | S | Channel | PT | **Downstream-layer reconstruction** | Yes | No (activations) | Prune | LASSO selection + least-squares weight reconstruction | Joint sparse regression |
| Network Slimming (Liu et al., 2017) | 2017 | S | Channel | Train | Task loss + sparsity of BN scaling factors | Yes | Yes at selection (BN scales) | Prune | Fine-tune resulting compact network | Fixed ranking |
| Taylor Pruning (Molchanov et al., 2016) | 2017 | S | Filter / feature map | PT | Approximate change in task loss | Yes | No (data/gradients) | Prune | Interleaved fine-tuning | Iterative re-scoring |
| Soft Filter Pruning (SFP) (He et al., 2018a) | 2018 | S | Filter | Train | Filter norm + task objective | Joint | Yes for ranking | Prune | Pruned filters continue to be updated during training | Iterative ranking |
| AMC (He et al., 2018c) | 2018 | S | Channel | PT / search | Accuracy–resource trade-off | Yes | No (validation data) | Prune | Fine-tune selected compressed architecture | RL / architecture search |
| NISP (Yu et al., 2017) | 2018 | S | Neuron | PT | **Reconstruction of important responses in a later layer** | Yes | No (data) | Prune | Fine-tune after importance propagation | Propagated fixed scores |
| Discrimination-Aware Channel Pruning (Zhuang et al., 2018) | 2018 | S | Channel | PT | Downstream reconstruction + discriminative loss | Yes | No (labeled data) | Prune | Weight optimization + fine-tuning | Greedy / joint |
| FPGM (He et al., 2018b) | 2019 | S | Filter | PT | Weight-space redundancy / geometric median | Yes | Yes (weights only) | Prune | Fine-tune surviving network | Relational fixed score |
| Importance Estimation for Neural Network Pruning (Molchanov et al., 2019) | 2019 | S | Neuron / filter | PT | Approximate change in final task loss | Yes | No (data/gradients) | Prune | Iterative pruning and fine-tuning | Adaptive / iterative |
| MetaPruning (Liu et al., 2019) | 2019 | S | Channel | PT / search | Accuracy under FLOP / architecture constraint | Optional | No (training/validation data) | Prune | Weights generated by meta-network; final tuning possible | Architecture search |
| HRank (Lin et al., 2020) | 2020 | S | Filter | PT | Rank / information content of feature maps | Yes | No (activations) | Prune | Fine-tune surviving network | Fixed ranking |
| Only Train Once (OTO) (Chen et al., 2021) | 2021 | S | Parameter groups | Train | Task loss + structured group sparsity | No extra FT | No (training data) | Prune | Model and sparsity optimized jointly | Joint optimization |
| CoFi (Xia et al., 2022) | 2022 | S | Layers / heads / FFN / hidden units | Fine-tune | Task loss + knowledge distillation | Joint | No (task data) | Prune | Layerwise distillation and mask optimization | Learned joint masks |
| DepGraph (Fang et al., 2023) | 2023 | S | Coupled parameter groups | PT | Depends on importance criterion; norm used as baseline | Usually yes | Depends on criterion | Prune | Coupled structures removed consistently; usually fine-tuned | Dependency groups + score |
| **Reconstruction, compensation, merging, and subset selection** | | | | | | | | | | |
| Neuron Merging: Compensating for Pruned Neurons (Kim et al., 2020) | 2020 | S | Neuron / filter | PT | Feature-map preservation through neuron redundancy | No | Yes | Merge | Scaling matrix absorbed into the following layer | Pairwise / merge-based |

*Continued on next page*

*Table 3 continued*

| Paper / Method | Year | Type | Pruned object | Stage | Preserved / optimized quantity | Fine-tune? | Data-free selection? | Action | Adjustment after removal | Selection coupling |
|---|---|---|---|---|---|---|---|---|---|---|
| Data-Efficient Structured Pruning via Submodular Optimization (Halabi et al., 2022) | 2022 | S | Neuron / channel | PT | **Downstream-layer representation reconstruction** | No | No (limited unlabeled data) | Prune | Optimizes weights of the following layer | Greedy set-dependent |
| Optimal Brain Compression (OBC) (Frantar & Alistarh, 2022) | 2022 | U / semi-S | Weight | PT | **Layer-output reconstruction** | No | No (calibration data) | Prune | Exact OBS-style updates of surviving weights | Adaptive / sequential |
| **Modern language-model and post-training pruning** | | | | | | | | | | |
| SparseGPT (Frantar & Alistarh, 2023) | 2023 | U / semi-S | Weight | PT | **Layer-output reconstruction** | No | No (calibration data) | Prune | Sequential second-order updates to unpruned weights | Adaptive / sequential |
| LLM-Pruner (Ma et al., 2023) | 2023 | S | Coupled neurons / heads / channels | PT | Gradient-based functional / task sensitivity | Yes (LoRA) | No (small calibration set) | Prune | LoRA recovery of remaining model | Group-dependent |
| ZipLM (Kurtic et al., 2023) | 2023 | S | Layers / heads / FFN structures | PT / gradual | Loss–runtime trade-off and local reconstruction | Optional | No (calibration/task data) | Prune | Supports one-shot mode or iterative fine-tuning | Iterative / runtime-aware |
| Wanda (Sun et al., 2024) | 2024 | U / semi-S | Weight | PT | Activation-aware magnitude proxy | No | No (calibration activations) | Prune | None | Fixed score |
| FLAP (An et al., 2024) | 2024 | S | Channel | PT | **Recoverability of downstream output features** | No | No (calibration activations) | Prune | Bias compensation using baseline feature values | Score + global search |
| OSSCAR (Meng et al., 2024) | 2024 | S | Neuron / channel / attention structure | PT | **Consumer-layer representation reconstruction** | No | No (calibration data) | Prune | Optimizes surviving weights after structural removal | Joint / local combinatorial |
| HOPE (Mobahi & Bartlett, 2026) | 2026 | S | Neuron / block | PT | **Functional/operator-space representation** | Optional | Yes | Both | Low-rank projection / merging updates remaining operators | Joint subspace projection |

**Notation and interpretation.** S and U denote structured and unstructured pruning, respectively. PT denotes post-training pruning. "Fine-tune?" refers to a recovery stage after pruning; "Joint" means that sparsification is itself performed during training or fine-tuning. "Data-free selection" asks whether examples are required specifically to determine what is removed or merged. A "fixed score" means that units are ultimately selected using scalar per-unit scores; it does *not* imply that the true pruning distortion is mathematically additive. Adaptive, greedy, or joint methods explicitly make the decision for one unit depend, to varying degrees, on other pruning decisions. *Pruning-at-initialization methods subsequently train the resulting sparse network. †Lottery-ticket discovery itself requires training the dense network, although the pruning decision is based on the resulting weights.

## B CONSOLIDATION RULE

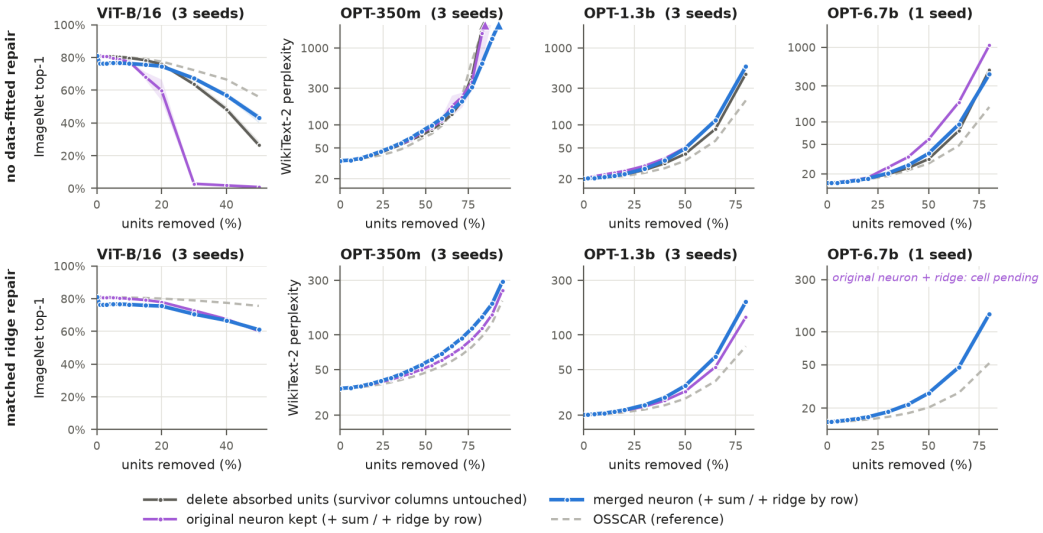


Figure 5: Evaluation of merged neuron consolidation (weighted average), original neuron selection, and deleting absorbed units on ViT-B/16 and OPT models. The top row shows performance without data-fitted repair, while the bottom row applies matched ridge repair, benchmarked against OSSCAR.

## C Derivation of ( 4)

For $\ell \in \{i, j\}$, expand around the weighted average neuron:

$$\begin{aligned}\|h_\ell - q_{ij}\|_P^2 = \|h_\ell - \mu_{ij}\|_P^2 + \|\mu_{ij} - q_{ij}\|_P^2 \\ + 2\langle h_\ell - \mu_{ij}, \mu_{ij} - q_{ij}\rangle_P.\end{aligned} \tag{12}$$

Now, we expand equation 2:

$$\begin{aligned}\mathcal{E}_{\text{pair}}(i, j; q_{ij}) &= a_i\|h_i - q_{ij}\|_P^2 + a_j\|h_j - q_{ij}\|_P^2 \\ &= a_i\|h_i - \mu_{ij}\|_P^2 + a_i\|\mu_{ij} - q_{ij}\|_P^2 + a_j\|h_j - \mu_{ij}\|_P^2 + a_j\|\mu_{ij} - q_{ij}\|_P^2 \\ &+ 2a_i\langle h_i - \mu_{ij}, \mu_{ij} - q_{ij}\rangle_P + 2a_j\langle h_j - \mu_{ij}, \mu_{ij} - q_{ij}\rangle_P \\ &= \frac{a_i a_j}{a_i + a_j}\|h_i - h_j\|_P^2 + (a_i + a_j)\|\mu_{ij} - q_{ij}\|_P^2 \\ &+ \underbrace{2\langle a_i(h_i - \mu_{ij}) + a_j(h_j - \mu_{ij}), \mu_{ij} - q_{ij}\rangle_P}_{0} \\ \mathcal{E}_{\text{pair}}(i, j; q_{ij}) &= \frac{a_i a_j}{a_i + a_j}\|h_i - h_j\|_P^2 + (a_i + a_j)\|\mu_{ij} - q_{ij}\|_P^2\end{aligned}$$