# SUBSPACE NODE PRUNING: RETRAINING-FREE NEURAL NETWORK COMPRESSION VIA ORTHOGONAL SUBSPACE PROJECTIONS

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

## ABSTRACT

Node pruning is the art of removing computational units such as neurons, filters, attention heads, or even entire layers to significantly reduce inference time while retaining network performance. In this work, we show that a broad family of importance measures, from activation-based to Fisher-based, share a common quadratic form, and that through a lower-triangular change of basis we can reach a pruning subspace in which whole nodes can be pruned with a simultaneous recovery of the impact of lost units via linear least squares. Furthermore, the importance measures within our subspace are decoupled and additive through an orthogonalization of the layer activations, allowing multi-node pruning within layers without interference between importance scores. Our method, Subspace Node Pruning (SNP), outperforms the compared baselines on VGG-16 and ResNet-50 on ImageNet. On OPT-125M and LLaMA-2 7B, it matches the state-of-the-art LLM Surgeon method at a 40-shot schedule and outperforms it in every case where fewer shots are used. All of this is achieved without any retraining after pruning.

## 1 INTRODUCTION

Alongside hardware optimizations (Jouppi et al., 2018; Choquette et al., 2021), software frameworks (Abadi et al., 2015; Bradbury et al., 2018; Paszke et al., 2019), and quantization (Krishnamoorthi, 2018; Gholami et al., 2022), exist methods which directly modify and compress neural network models to reduce computational cost while maintaining accuracy.

Neural network pruning is a particular approach which aims to remove parameters from a trained model in order to reduce the computational execution (inference) time while maintaining its performance. Unfortunately, unstructured approaches to pruning can result in models with arbitrarily sparsified weight matrices whose multiplication cannot easily be accelerated at compute time. It is therefore desirable to prune structured aspects of neural networks models: whole nodes, convolutional filters, transformer heads, or other structured groups of parameters. Herein, we refer to any of these sub-parts of networks as network 'units'. Two broad approaches have emerged for network pruning. Pruning is either applied to pre-trained networks while trying to maintain their performance, or is carried out as an iterative process during training. We limit our investigation to the pruning of pre-trained networks, assuming that one starts out with a well-performing model and does not wish to carry out any further (re)training.

**Importance Scores** When choosing which units of a network to prune, there must first be an attribution of the relative 'importance' of each network unit. A number of methods use the magnitude of weights as a proxy for an importance score, assuming that smaller weights have a smaller impact on a network's computation (Li et al., 2016). Methods of greater complexity are data-driven, making use of training data for forward, and in some cases backward, passes to enhance importance measures. The simplest of these score units by their activations alone (Hu et al., 2016; Berisha et al., 2025), or by their activations combined with their outgoing weights (He et al., 2017; Luo et al., 2017; Sun et al., 2023). Loss-based measures instead use a $1^{\text{st}}$-order Taylor expansion of the loss (Molchanov et al., 2016; 2019), Fisher information (Theis et al., 2018), or even downstream network importances (Yu et al., 2018). $2^{\text{nd}}$-order measures date back to Optimal Brain Damage and Optimal Brain Surgeon (LeCun et al., 1989; Hassibi & Stork, 1992), and have been made tractable



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.

for large networks by Kronecker-factored curvature approximations (Martens & Grosse, 2015; Wang et al., 2019), more recently for structured pruning of large language models in LLM Surgeon (van der Ouderaa et al., 2024).

Other approaches look at feature maps to determine filters which have greatest task-relevant information (Liu et al., 2023; Zhang et al., 2022a) or various correlation measures which treat high correlation between units as a sign of redundancy (Mariet & Sra, 2015; Ayinde et al., 2019; Cuadros et al., 2020; Kim et al., 2020; Goldberg et al., 2022; Zhang et al., 2022a; He et al., 2019).

**Reconstruction** After having pruned a network, it is often possible to analytically determine specific changes which can be made to the remaining parameters in a network to account for the removed unit. We refer to this process as 'reconstruction'. Unlike retraining or finetuning, reconstruction is a single step modification to a network. Optimal Brain Surgeon (Hassibi & Stork, 1992) derives such a change in closed form from the curvature of the loss, an approach that LLM Surgeon (van der Ouderaa et al., 2024) scales to structured pruning of large language models. When several units are removed at once, however, their costs and compensations interact. Optimal Brain Compression (Frantar & Alistarh, 2022) and ZipLM (Kurtić et al., 2023) therefore remove units greedily, one at a time, updating the inverse curvature after each removal so that later costs account for earlier ones.

Choosing which nodes to prune and which can be best reconstructed is also an important consideration. Mariet & Sra (2015) identified redundant nodes by using determinantal point processes, pruned these nodes, and recovered their impact on a network by linear least squares (LLS). Similarly, He et al. (2017) selected nodes based on LASSO regression and again used LLS to approximate pruned nodes. Chee et al. (2022) instead select the units to keep by pivoted QR, an interpolative decomposition, and reconstruct the remaining units from them by least squares. Other work uses less expressive approaches, with single scalar values used for reconstruction per node (Luo et al., 2017), or even attempt data-free approaches to reconstructing unit activity (Theus et al., 2024). Other work goes further still, into a non-linear least squares solution attempt using evolutionary algorithms (Chin et al., 2018), or application of approximate reconstruction even to large language models (Frantar & Alistarh, 2023; Li et al., 2024), for instance by compensating removed channels through bias terms (An et al., 2024).

**Transformer Pruning** Methods in transformer model pruning go beyond estimating local or global importance, often pruning different granularities of nodes simultaneously. Zheng et al. (2022) designed an optimization problem to prune attention heads, embedding dimensions and the hidden layers of the feedforward networks. They combined a $2^{\text{nd}}$-order Taylor estimation with evolutionary algorithms to optimize a pruning mask. Yang et al. (2023b) went even further by pruning *query* and *key* embedding dimensions independent of the *value* embedding dimension. Several other recent influential methods train networks for estimating global sparsity (Yu & Xiang, 2023; Yang et al., 2023a; Yu et al., 2022) and therefore fall outside of the scope of our setting. Dynamic removal of tokens/patches to reduce the inference time for a given sample is also an area under increasing investigation (Rao et al., 2021; Song et al., 2022; Wang et al., 2022; Liang et al., 2022) but this prunes inputs rather than the network directly.

**This work** Herein we describe and empirically validate a novel framework for the entire pipeline of node importance scoring, pruning, and reconstruction. Our core contributions are:

1. We provide a common form for a range of importance measures. Activation-based (zeroth-order), first-order, and second-order importance measures share a common quadratic form that differs only in a layer-wise sensitivity. The importance measure is therefore an interchangeable component, and the properties below hold for all of them.
2. We show how the orthogonalization of unit activations makes importance scores decoupled and additive. By using a lower-triangular orthogonalization transform, each unit's score only reflects the importance not already captured by the units preceding it. Combined with a unit ordering method, multiple units can then be pruned within a layer while the total 'lost importance' is exactly the sum of their individual importances.
3. We construct a subspace in which pruning is carried out linearly reconstructs lost activity, and is equivalent to pruning network units. This differentiates our subspace from other subspace methods (like PCA, ZCA etc) in that pruning in our subspace reduced network layer widths and provides measurable efficiency gains.

Existing methods either score units independently, neglecting correlations between importance scores (van der Ouderaa et al., 2024), or account for these correlations by removing units sequentially (Frantar & Alistarh, 2022; Kurtić et al., 2023), which requires a separate update after every removal. In comparison, our subspace has exactly additive pruning costs allowing multi-unit pruning while taking into account correlations, and is efficient in not requiring a re-computation of the subspace if the number of nodes being pruned is changed. We demonstrate the effectiveness and computational efficiency of our approach in comparison to the state of the art by application to ImageNet-trained VGG-16 and ResNet-50, as well as OPT and LLaMA-2 language models evaluated on the WikiText dataset, all without any retraining.

## 2 METHODS

Consider a typical deep neural network (DNN) architecture, in which the outputs at each layer, $l \in \{1, \dots, L\}$, are defined as $\mathbf{X}_l = f_l(\mathbf{Y}_l) = f_l(\mathbf{W}_{l-1}\mathbf{X}_{l-1})$, where $\mathbf{X}_l \in \mathbb{R}^{n_l \times s}$ is a matrix of outputs for layer $l$ consisting of $n_l$ units for $s$ samples. These layer outputs are composed based upon a matrix multiplication of the previous layer outputs, from weights $\mathbf{W}_l \in \mathbb{R}^{n_{l+1} \times n_l}$, and by an element-wise transfer function $f_l(\cdot)$. We denote by $\mathcal{L}$ the loss of the network on these $s$ samples, summed over samples. We consider these fully-connected deep neural networks to introduce our approach, however, we also apply this theory to convolutional and transformer architectures in the results that follow.

### 2.1 FACTORIZING NEURAL CONTRIBUTIONS

Importance scores, whether based upon node activation or loss impact via first/second-order Taylor approximation, are approximate (or heuristic) measures of the degree of impact that removing an individual unit has on a network's performance. A broad set of importance scores can be expressed in the form that follows, letting $\mathbf{x}_l^{(k)}$ be the activity of all units at layer $l$ for sample $k$, we define the importance of all units in a layer as

$$I_l = \sum_{k=1}^{s} (\mathbf{A}_l^{(k)} \mathbf{x}_l^{(k)})^\top (\mathbf{A}_l^{(k)} \mathbf{x}_l^{(k)}),$$

where $\mathbf{A}_l^{(k)}$ is the sample-wise sensitivity matrix that maps the unit activities onto the space in which their impact is measured for importance. The choice of $\mathbf{A}_l^{(k)}$ recovers commonly used importance measures as shown in Table 1.

Throughout, we use a single sensitivity shared across samples, $\mathbf{S}_l = \frac{1}{s} \sum_k \mathbf{A}_l^{(k)\top} \mathbf{A}_l^{(k)}$, so that the layer importance, in matrix form, is

$$I_l = \operatorname{tr}\left(\mathbf{X}_l^\top \mathbf{S}_l \mathbf{X}_l\right) = \operatorname{tr}\left(\mathbf{S}_l \mathbf{C}_l\right), \qquad \mathbf{C}_l = \mathbf{X}_l \mathbf{X}_l^\top, \tag{1}$$

with per-unit importances given by the diagonal terms, $I_{l,i} = [\mathbf{S}_l]_{ii}[\mathbf{C}_l]_{ii}$. For the activation-based measures and K-FAC-based curvature estimation method the sensitivity does not depend on the sample and this form is exact. For the other loss-based measures, activities and sensitivities are averaged separately, which yields the K-FAC form $\mathbf{S}_l = \frac{1}{2}\mathbf{W}_l^\top \bar{\mathbf{H}}_l \mathbf{W}_l$, with $\bar{\mathbf{H}}_l$ the sample average of the chosen curvature estimate (Martens & Grosse, 2015; Wang et al., 2019); for the first-order measure, this is the empirical Fisher up to a factor of $\frac{1}{2}$.

A typical pruning pipeline removes the $m$ units of lowest importance score, together with their outgoing weights. However, these scores are typically then used per node, individually, as if each node's contribution to this total layer importance were independent and as if $\mathbf{C}_l$ were diagonal. In practice, units are often highly redundant in their activity, such that when a unit is removed, much of its contribution could be carried by the units that remain, given appropriate adjustment of their outgoing weights. In the following, we instead transform the layer so that shared importance is removed from the units to be pruned before any pruning takes place.

Table 1: Common importance measures expressed in terms of the activities at layer $l$, transformed by a sample-wise sensitivity matrix $\mathbf{A}_l^{(k)}$. Here $\mathbf{g}_l^{(k)} = \partial \mathcal{L} / \partial \mathbf{x}_l^{(k)}$ is the gradient of the loss with respect to the unit activities, and $\mathbf{H}_l^{(k)}$ is a curvature estimate (Martens, 2020) of the loss with respect to the layer's output $\mathbf{y}_{l+1}^{(k)}$: the Gauss–Newton Hessian, the true Fisher, the empirical Fisher $\delta_l^{(k)} \delta_l^{(k)\top}$ with $\delta_l^{(k)} = \partial \mathcal{L} / \partial \mathbf{y}_{l+1}^{(k)}$, or its sample average $\bar{\mathbf{H}}_l$ as in K-FAC (Martens & Grosse, 2015). Since $\mathbf{g}_l^{(k)} = \mathbf{W}_l^\top \delta_l^{(k)}$, the empirical Fisher recovers the first-order measure up to a factor of $\frac{1}{2}$. The per-unit importance $I_{l,i}$ is the contribution of unit $i$ to the layer-wide importance $I_l$, excluding cross terms between units, where $x_{l,i}^{(k)}$ and $g_{l,i}^{(k)}$ are the $i$-th elements of $\mathbf{x}_l^{(k)}$ and $\mathbf{g}_l^{(k)}$, and $\mathbf{w}_{l,i}$ is the $i$-th column of $\mathbf{W}_l$. Citations indicate methods using these measures or close variants thereof.

| | | |
|---|---|---|
| **Squared input activation** (Hu et al., 2016; Berisha et al., 2025) | | |
| $\mathbf{A}_l^{(k)}$: $\mathbf{I}$ | $I_l$: $\sum_k \mathbf{x}_l^{(k)\top} \mathbf{x}_l^{(k)}$ | $I_{l,i}$: $\sum_k \left(x_{l,i}^{(k)}\right)^2$ |
| **Squared (layer) output activation** (He et al., 2017; Luo et al., 2017; Sun et al., 2023) | | |
| $\mathbf{A}_l^{(k)}$: $\mathbf{W}_l$ | $I_l$: $\sum_k \mathbf{x}_l^{(k)\top} \mathbf{W}_l^\top \mathbf{W}_l \mathbf{x}_l^{(k)}$ | $I_{l,i}$: $\lVert \mathbf{w}_{l,i} \rVert^2 \sum_k \left(x_{l,i}^{(k)}\right)^2$ |
| **First-order Taylor** (Molchanov et al., 2016; Theis et al., 2018; Molchanov et al., 2019) | | |
| $\mathbf{A}_l^{(k)}$: $\mathbf{g}_l^{(k)\top}$ | $I_l$: $\sum_k \left(\mathbf{g}_l^{(k)\top} \mathbf{x}_l^{(k)}\right)^2$ | $I_{l,i}$: $\sum_k \left(g_{l,i}^{(k)} x_{l,i}^{(k)}\right)^2$ |
| **Second-order Taylor / Fisher** (LeCun et al., 1989; Hassibi & Stork, 1992; Wang et al., 2019; Yang et al., 2023b; van der Ouderaa et al., 2024) | | |
| $\mathbf{A}_l^{(k)}$: $\left(\mathbf{H}_l^{(k)}/2\right)^{1/2} \mathbf{W}_l$ | $I_l$: $\frac{1}{2} \sum_k \mathbf{x}_l^{(k)\top} \mathbf{W}_l^\top \mathbf{H}_l^{(k)} \mathbf{W}_l \mathbf{x}_l^{(k)}$ | $I_{l,i}$: $\frac{1}{2} \sum_k \left(x_{l,i}^{(k)}\right)^2 \mathbf{w}_{l,i}^\top \mathbf{H}_l^{(k)} \mathbf{w}_{l,i}$ |

### 2.2 Removing redundant shared importances

Prior to pruning, we propose to transform the unit activities of a layer into a latent basis, $\hat{\mathbf{X}}_l = \mathbf{M}_l \mathbf{X}_l$, with an invertible matrix $\mathbf{M}_l$. Defining the transformed weights and sensitivities

$$\tilde{\mathbf{W}}_l = \mathbf{W}_l \mathbf{M}_l^{-1}, \qquad \tilde{\mathbf{A}}_l^{(k)} = \mathbf{A}_l^{(k)} \mathbf{M}_l^{-1},$$

we have $\tilde{\mathbf{W}}_l \hat{\mathbf{X}}_l = \mathbf{W}_l \mathbf{X}_l$ and $\tilde{\mathbf{A}}_l^{(k)} \hat{\mathbf{x}}_l^{(k)} = \mathbf{A}_l^{(k)} \mathbf{x}_l^{(k)}$, so that both the network output and the layer importance $I_l$ are unchanged for any choice of $\mathbf{A}_l^{(k)}$.

In the latent basis, the layer importance reads

$$I_l = \operatorname{tr}\left(\tilde{\mathbf{S}}_l \mathbf{M}_l \mathbf{C}_l \mathbf{M}_l^\top\right) = \sum_j \hat{I}_{l,j} + \sum_{j \neq j'} \left[\tilde{\mathbf{S}}_l\right]_{jj'} \left[\mathbf{M}_l \mathbf{C}_l \mathbf{M}_l^\top\right]_{jj'}, \tag{2}$$

where $\tilde{\mathbf{S}}_l = \mathbf{M}_l^{-\top} \mathbf{S}_l \mathbf{M}_l^{-1}$, and $\hat{I}_{l,j} = [\tilde{\mathbf{S}}_l]_{jj} [\mathbf{M}_l \mathbf{C}_l \mathbf{M}_l^\top]_{jj}$ takes the same form as the per-unit importances $I_{l,i}$. If the cross terms vanish, the latent importances sum exactly to $I_l$, each equals the change in $I_l$ caused by removing that latent unit, and the impact of removing any set of latent units is the sum of their importances. *Our goal is therefore to choose $\mathbf{M}_l$ such that all cross terms vanish and that each latent unit independently contributes to the total layer importance.*

Many transformations could remove these cross terms, for example orthogonalizing transforms such as PCA or ZCA. However, pruning in such bases gives a low-rank weight matrix rather than removing any units (Levin et al., 1993). To remove units, the latent units that are kept must depend only on the original units that are kept. Requiring this for pruning the last $n_l - m$ latent units, for every $m$, restricts $\mathbf{M}_l$ to be lower-triangular. Then, with $\mathcal{K} = \{1, \ldots, m\}$ the kept units, $\hat{\mathbf{X}}_{l,\mathcal{K}} = (\mathbf{M}_l)_{\mathcal{K}\mathcal{K}} \mathbf{X}_{l,\mathcal{K}}$, and pruning the last $n_l - m$ latent units removes the corresponding original units (Figure 1). Note, this setup assumes that our nodes are somehow already ordered such that the last nodes in our original space are the ones we wish to prune first. We address how to choose this ordering in Section 2.3.

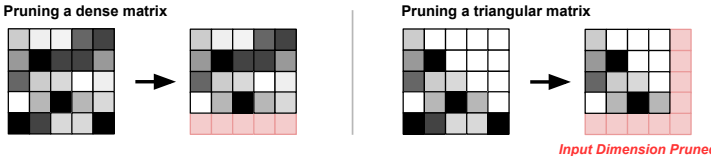



Figure 1: Why the subspace must be constructed with a lower-triangular matrix. Left: with a dense $\mathbf{M}_l$, removing latent units only lowers the rank of the layer (number of nodes in the latent space), and every original input unit remains in use. Right: with a lower-triangular $\mathbf{M}_l$, removing the bottom row of $\mathbf{M}_l$ removes the last original unit.

So, can we find a lower-triangular $\mathbf{M}_l$ that removes the cross terms? By equation 2, they vanish whenever $\mathbf{M}_l\mathbf{C}_l\mathbf{M}_l^\top = \mathbf{D}_l$ is diagonal, regardless of $\mathbf{S}_l$. For lower-triangular $\mathbf{M}_l$, which we take to have a unit diagonal without loss of generality (rescaling a latent unit changes neither the output nor its importance), rearranging $\mathbf{M}_l\mathbf{C}_l\mathbf{M}_l^\top = \mathbf{D}_l$ gives $\mathbf{C}_l = \mathbf{M}_l^{-1}\mathbf{D}_l\mathbf{M}_l^{-\top}$. This is the LDL decomposition of $\mathbf{C}_l$ with unit lower-triangular factor $\mathbf{M}_l^{-1}$, which is unique for a given ordering of the units when $\mathbf{C}_l$ is positive definite. Equivalently, $\mathbf{M}_l$ performs a Gram–Schmidt orthogonalization of the unit activities: latent unit $j$ is the residual of original unit $j$ after regressing out units $1, \ldots, j-1$. The last latent units, which are pruned first, thus retain only activity that cannot be explained by earlier units, and the diagonal of the matrix above gives their importances, $\hat{I}_{l,j} = D_{l,jj}[\tilde{\mathbf{S}}_l]_{jj}$.

Pruning then proceeds from the last latent unit backwards, i.e. by removing rows of $\mathbf{M}_l$ from the bottom. With $\mathcal{K} = \{1, \ldots, m\}$ the kept and $\mathcal{P} = \{m+1, \ldots, n_l\}$ the pruned units, the kept latent activities are $\hat{\mathbf{X}}_{l,\mathcal{K}} = (\mathbf{M}_l)_{\mathcal{K}\mathcal{K}}\mathbf{X}_{l,\mathcal{K}}$, so that the pruned layer acts directly on the original kept units, with weights

$$\hat{\mathbf{W}}_l = (\tilde{\mathbf{W}}_l)_{:\mathcal{K}}(\mathbf{M}_l)_{\mathcal{K}\mathcal{K}} = (\mathbf{W}_l)_{:\mathcal{K}} + (\mathbf{W}_l)_{:\mathcal{P}}\,(\mathbf{C}_l)_{\mathcal{P}\mathcal{K}}(\mathbf{C}_l)_{\mathcal{K}\mathcal{K}}^{-1}. \tag{3}$$

The second term is the least-squares reconstruction of the pruned units' activities from the kept ones, as used for pruning by Mariet & Sra (2015) and He et al. (2017). Here, however, it is not a separate step: as the first equality shows, it follows directly from truncating the lower-triangular basis, such that pruning and reconstruction become a single operation. The corresponding rows of $\mathbf{W}_{l-1}$ are removed. The kept units take over all contributions of the pruned units that they can linearly explain, and only the residual activity is discarded. Since the cross terms vanish, the layer importance decreases by exactly $\sum_{j\in\mathcal{P}} \hat{I}_{l,j}$, such that the importance lost after reconstruction is exactly what was known before pruning. The latent importances computed before pruning are thus exact predictions of the importance lost after reconstruction, unlike standard per-unit scores, which ignore both the cross terms and the compensation by the kept units. The kept latent units thus span the subspace of the original activities from which our method takes its name: for the kept units, it retains the largest possible share of the layer importance, whatever the importance measure. In practice, $\mathbf{C}_l$ may be singular, for example due to inactive or duplicated units, and one could therefore add a small damping $\lambda\mathbf{I}$ before the decomposition.

### 2.3 Choosing a pruning order

The method above assumes that units are ordered from most to least important, such that those to be pruned first come last in the layer. To minimize the importance lost by pruning, we seek the permutation of the units for which the pruned latent units carry the least total importance, $\min \sum_{j\in\mathcal{P}} \hat{I}_{l,j}$. Finding this ordering is a combinatorial problem. We avoid it by scoring each unit once, before the factorization, with a score that does not depend on the order of the other units. When pruning in several shots, the scores are recomputed at the start of each shot.

Ideally, each unit would be scored by the importance which cannot be carried by the other units. Placing unit $i$ last, its latent activity is the residual after regressing out all other units, with $D_{l,n_l n_l} = 1/[\mathbf{C}_l^{-1}]_{n_l n_l}$, and since the last column of $\mathbf{M}_l^{-1}$ is the unit vector, $[\tilde{\mathbf{S}}_l]_{n_l n_l} = [\mathbf{S}_l]_{ii}$. We therefore score each unit by its latent importance as if it had been placed last,

$$I_{l,i}^{\perp} = \frac{[\mathbf{S}_l]_{ii}}{[\mathbf{C}_l^{-1}]_{ii}}, \tag{4}$$

**Algorithm 1** Pruning a network in the latent basis

**Require:** weights $\{\mathbf{W}_l\}$, calibration data, damping $\lambda$, number of shots $T$, fraction $\tau$ or FLOPs target $F$

**for** shot $t = 1, \ldots, T$ **do**
  Compute $\{\mathbf{X}_l\}$, $\{\mathbf{S}_l\}$ on the current network
  **for** each layer $l$ **do**
    $\mathbf{C}_l \leftarrow \mathbf{X}_l \mathbf{X}_l^\top$ ▷ if necessary adding $\lambda \mathbf{I}$
    Reorder units by decreasing $I^{\perp}_{l,i}$ ▷ equation 4
    $\mathbf{L}_l \mathbf{D}_l \mathbf{L}_l^\top \leftarrow \mathbf{C}_l$, $\hat{I}_{l,j} \leftarrow D_{l,jj}[\mathbf{L}_l^\top \mathbf{S}_l \mathbf{L}_l]_{jj}$ ▷ LDL, $\mathbf{L}_l = \mathbf{M}_l^{-1}$
  **end for**
  Select trailing units $\mathcal{P}_l$ per layer using $\tau$ or the shot's target $F_t$ ▷ Sec. 2.4
  **for** each layer $l$ **do**
    Update $\mathbf{W}_l$ by equation 3; remove rows $\mathcal{P}_l$ of $\mathbf{W}_{l-1}$
  **end for**
**end for**

and order the units by decreasing $I^{\perp}_{l,i}$. Note that this heuristic allows us to avoid the combinatorial problem of checking all possible orders of nodes.

Compared with $I_{l,i} = [\mathbf{S}_l]_{ii}[\mathbf{C}_l]_{ii}$, the activity $[\mathbf{C}_l]_{ii}$ is replaced by $1/[\mathbf{C}_l^{-1}]_{ii} \leq [\mathbf{C}_l]_{ii}$, with equality only if unit $i$ is uncorrelated with all others, so that the importance of a node which is also shared with other nodes is discounted. For the K-FAC measure, equation 4 is the Optimal Brain Surgeon cost of removing an input unit with optimal compensation, as used in LLM Surgeon (Hassibi & Stork, 1992; van der Ouderaa et al., 2024). The LDL decomposition in the resulting order then removes all remaining cross terms, as described above.

### 2.4 Pruning full networks

The importances above determine which units to prune within a layer, but not how to distribute pruning across the layers of a network. For this, we use that the factorization provides the latent importances $\hat{I}_{l,j} = D_{l,jj}[\tilde{\mathbf{S}}_l]_{jj}$ without extra computation, and that, since the cross terms vanish, pruning a set of trailing latent units removes exactly the sum of their importances. How these are compared across layers depends on the importance measure.

For the activation-based measures, importances are not comparable across layers, and we instead compare fractions of each layer's importance. Pruning the latent units $i, \ldots, n_l$ removes the fraction $\rho_{l,i} = \frac{1}{I_l} \sum_{j=i}^{n_l} \hat{I}_{l,j}$ of the layer importance. We set a single global parameter $\tau$, the fraction of importance to be removed from every layer, and in each layer prune the largest set of trailing units with $\rho_{l,i} \leq \tau$. For the squared input activation ($\mathbf{S}_l = \mathbf{I}$), $\hat{I}_{l,j}$ reduces to the latent variance $D_{l,jj}$, and $\rho_{l,i}$ to the fraction of activity variance removed.

For the loss-based measures, the latent importances estimate the increase in loss caused by pruning, and are directly comparable across layers. We therefore prune globally, as in van der Ouderaa et al. (2024): at each step, among the trailing latent units of all layers, we prune the one with the smallest importance, until the target FLOPs reduction is reached. Within a layer, the importance lost is exactly the sum of the pruned latent importances, while across layers this sum neglects interactions between layers. The full pipeline is given in Algorithm 1.

## 3 Experimental Setup

**Networks and datasets** To demonstrate the efficacy of our proposed method, we first apply it to the single-branch models VGG-16 (Simonyan & Zisserman, 2014), followed by ResNet-50 (He et al., 2016), and lastly the transformer architectures OPT (Zhang et al., 2022b), and LLaMA-2 7B (Touvron et al., 2023). OPT and LLaMA-2 are evaluated on the WikiText dataset (Merity et al., 2016). All other models are pretrained and evaluated on the ILSVRC 2012 (ImageNet) dataset (Russakovsky et al., 2015). The pruning procedure on LLMs was based entirely upon the openly available codebase of LLM Surgeon with matched hyperparameters, samples, sequence lengths, and sparsity

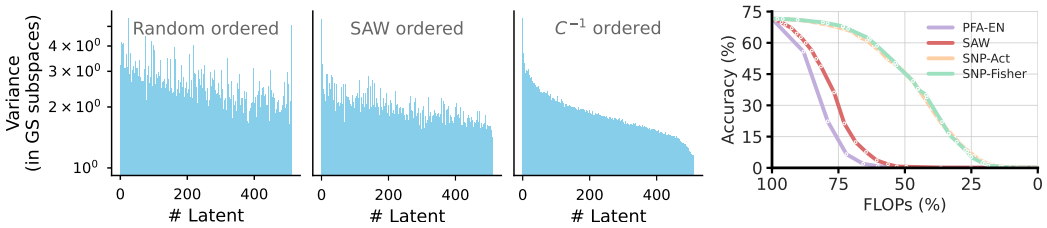
target computations. Modified code thereof and all code used to produce the results herein is available at <SEE ATTACHED ZIP>.

**Prunable units** For the vision models, we apply Algorithm 1 to every prunable layer, with the input units of each linear transform as the units to be pruned, as described in Section 2.1. For convolutional layers, the input is unfolded into patches, and an input channel then corresponds to a group of $k^2$ units, which are kept or pruned together. For the language models, we follow the structured pruning protocol of LLM Surgeon (van der Ouderaa et al., 2024), which prunes both the rows (outputs) and the columns (inputs) of every linear transform in the attention and MLP blocks, with token positions treated as samples. As in van der Ouderaa et al. (2024), for dimensions that are shared with the residual stream and between the query, key, and value projections, pruned rows and columns are set to zero rather than removed and we report sparsity as the fraction of parameters set to zero.

**Metrics and hardware** We measure performance as the Top-1 test accuracy for vision models and perplexity for language models, in relation to their FLOPs, and parameter count. We use the fvcore package (`https://github.com/facebookresearch/fvcore`) to estimate the FLOPs. All experiments were run on between one and four Nvidia A100 GPU machines.

# 4 Results

## 4.1 Pruning order




(a) Effect of node ordering (b) Accuracy after pruning

Figure 2: **Node ordering and pruning performance on VGG-16. (a)** Subspace importances $\hat{I}_{l,j}$ of layer 12, for units ordered randomly (left), by the summed absolute weights (SAW) of Li et al. (2016) (middle), and by our score $I^{\perp}_{l,i}$ (right). Units to the right are pruned first. **(b)** Accuracy over a range of pruning ratios for our method (SNP) and baseline methods (Appendix A). SNP-Act uses the squared input activation with the $I^{\perp}$ ordering, and SNP-Fisher is SNP with the empirical Fisher. PFA-EN is the only baseline that uses PCA to determine global importance.

Because our latent importances are additive, the importance lost by pruning is the sum over the trailing latent units, so a good ordering places the least important latent units last. Figure 2a shows that the ordering determines how well this is achieved. Randomly ordering units before transformation to our proposed subspace gives little relationship between the ordering of the latent bases and their importance measure. Ordering units by the summed absolute weight (SAW) importance heuristic (Li et al., 2016) before the subspace transformation improves on using a random order, but large latent importances are still distributed across the ordering. Ordering by $I^{\perp}_{l,i}$ (equation 4) instead gives latent importances that more smoothly reduce from first to last unit, such that the units pruned first carry the least importance. Although the ordering is obtained from a single scoring rather than a combinatorial search (Section 2.3), it nonetheless closely follows the redundancy captured by our importance measure and works well in practice.

## 4.2 Convolutional network prunings

**Single-branch networks** Figure 2b compares our method (SNP with activity vs Fisher derived importance scores) against baseline methods on VGG-16 without any finetuning (see Appendix A for baseline details). SNP retains most of the original accuracy, degrading only slowly as pruning

Table 2: **Pruning LLMs without retraining.** Perplexity on WikiText-2 after structured pruning of rows and columns with SNP and LLM Surgeon (van der Ouderaa et al., 2024), both with the empirical-Fisher K-FAC curvature, for different numbers of shots and sparsities (fraction of parameters removed). Lower is better; the better method in each pair is in bold, and – marks settings not evaluated. Results on OPT-125M for all numbers of shots are given in Table 4.

| | | OPT-125M | | LLaMA-2 7B | |
|---|---|---|---|---|---|
| **Shots** | **Sparsity** | SNP | LLM Surgeon | SNP | LLM Surgeon |
| Dense | | 27.65 | | 5.12 | |
| 1 | 10% | **28.48** | 29.30 | **5.89** | 7237 |
| | 30% | **55.13** | 3357 | **19.14** | 4791 |
| | 50% | **579.72** | 4258 | **241.77** | 13169 |
| 5 | 10% | **27.85** | 28.97 | 5.49 | **5.27** |
| | 30% | **32.41** | 33.54 | **8.27** | 12.13 |
| | 50% | **54.21** | 59.79 | **19.00** | 7986 |
| 40 | 50% | **48.89** | 49.72 | **14.88** | 15.67 |

increases. At 50% of FLOPs pruned, competing non-reconstructive methods have reached close to chance, where SNP methods maintain accuracies above 40%.

**Multi-branch networks** Figure 3 compares SNP with Intra-Fusion (IF) (Theus et al., 2024), a method which uses optimal transport as a mechanism for reconstruction, across pruning groups of ResNet-50. SNP outperforms IF for all groups, and continues to do so when $\mathbf{C}_l$ is estimated from only 1024 images or even white noise inputs. This demonstrates the efficacy of the linear least squares reconstruction embedded within our subspace construction.

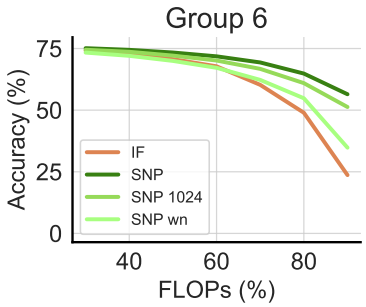



Figure 3: **SNP vs. Intra-Fusion for reconstruction.** A comparison of pruning different groups of layers of ResNet-50. We compare our SNP method with the squared input activation measure against Intra-Fusion (IF). SNP 1024 and SNP wn estimate $\mathbf{C}_l$ from 1024 images and from white noise inputs, respectively, to show robustness to the samples used. SNP's LLS reconstruction outperforms IntraFusions Optimal Transport approach over the range tested.

### 4.3 Language models

Table 2 compares SNP against LLM Surgeon (van der Ouderaa et al., 2024) on OPT-125M and LLaMA-2 7B. LLM Surgeon is run with its authors' own codebase, with the same calibration data as SNP. To verify this setup, we compare our 40-shot runs of LLM Surgeon with the values reported by van der Ouderaa et al. (2024) in Appendix C and show that these match. The degradation of LLM Surgeon at low shot-number is therefore a previously underdiscussed failure mode. Both methods use the same empirical-Fisher curvature estimation, unit ordering, and weight compensation, and prune over a number of shots, re-estimating the curvature and the activity correlations (importances) after each. They differ only in how pruning costs combine: LLM Surgeon selects units by their individual costs, whereas SNP selects them by latent importances that are exactly additive within a layer (Section 2.4).

Across all cases bar one, regardless of the number of shots allowed, SNP leads in maintaining lower perplexity. Over an even more extended set of 60 settings tested on OPT-125M SNP achieves the lower perplexity in 59, see Appendix B, Table 4. LLM Surgeon's per-unit costs neglect the correlations among units that are pruned together, an error that grows with the number of units pruned together. Thus, LLM Surgeon's error is mitigated with more shots explaining why its performance is robust only in the large-shot regime. Since the latent importances of SNP already account for activity correlations, SNP needs far fewer shots for a given perplexity, and each shot saved avoids a full re-estimation of the curvature.

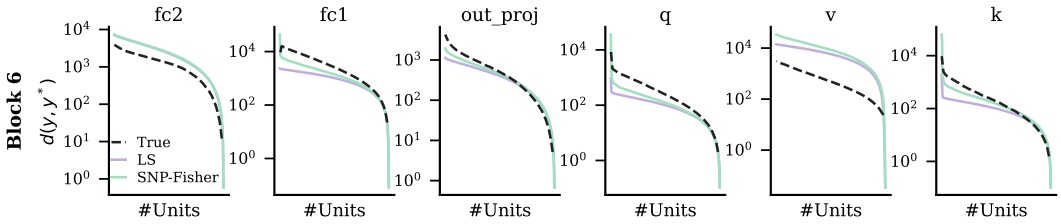



Figure 4: **True Loss Changes vs Loss Predictions of SNP vs LLM Surgeon.** These plots show the loss change induced by pruning units (more units pruned from right to left) using the SNP with empirical Fisher vs LLM Surgeon (LS) method. These loss magnitude changes are shown for the middle block of an OPT-125M model and for all of its linear layers including two fully connected (fc) layers of the MLP, an output projection layer (out_proj), as well as the Q, K, and V matrices.

To further illuminate the differences between SNP's (with empirical Fisher) and LLM Surgeon's importance measures, we plot these methods' predicted vs actual loss change for single shot pruning in the OPT-125M model. Figure 4 shows the impact of pruning inputs of the middle block of the OPT-125M model's fully connected (fc), output projecting (out_proj) and QKV matrices. As can be seen, SNP's predictions of loss difference for each node are equivalent to LLM Surgeon for the trailing units (where their measures are equivalent) but as one prunes more units (left to rigth), the impact of one node's pruning on other nodes increases and shows a deviation of our method vs LLM Surgeon. In five of the six plotted matrices, our deviation matches more closely to the true loss cost of pruning where LLM Surgeon underestimates it at high pruning ratios. More extended analysis of this is provided in Appendix C.1. Appendix C also provides further evidence of SNP's shot-level benefits against alternative LLM pruning methods such as SliceGPT (Ashkboos et al., 2024), K-OBD (LeCun et al., 1989; Wang et al., 2019), and LLM Surgeon.

## 5 Discussion

Our framework builds on and unifies several existing lines of work. Mariet & Sra (2015) and He et al. (2017) reconstructed pruned units by least squares as a separate step; in our subspace, this reconstruction follows directly from pruning within the subspace. K-FAC-based methods such as LLM Surgeon (van der Ouderaa et al., 2024) provide effective loss-based importances, which our framework adopts as one of many potential choices of sensitivity. However, where other methods score units individually, our subspace importances are exactly additive within a layer, such that the joint cost of pruning many correlated units is known. Our framework is also closely related to greedy sequential pruning with optimal compensation (Frantar & Alistarh, 2022; Kurtić et al., 2023). For a fixed order of removal, these sequential costs are precisely our latent importances and yet our importances are cheaper to compute.

Nonetheless, several questions remain open. First, the additivity of importances holds only within a layer. Pruning across layers introduces interactions between layers that our importances do not capture. Accounting for these interactions, for example through block-level curvature estimates, is a natural next step as well as giving block-level importances more focus. Second, we order units by a single-pass score, $I^{\perp}_{l,i}$ (equation 4), rather than solving the combinatorial problem of finding the optimal ordering. Greedy sequential methods (Frantar & Alistarh, 2022; Kurtić et al., 2023) choose the order adaptively instead, at the cost of an update per removed unit. Better single-step approximations of this ordering could aim to bridge the gap in computational cost and accuracy between these.

Overall, our results suggest that much of the error in structured pruning comes from treating the importances of correlated units as independent. Measuring importance in a basis where it is additive removes this error, reduces the number of costly curvature re-estimations needed to prune large models, and remains effective even when the correlations are estimated from small data samples. As importance measures continue to improve, we expect that computing them in such a subspace will remain a cheap and broadly applicable way to make them reliable when many units are pruned at once.

### AI USE STATEMENT

We have not used generative AI tools for any of the tasks listed as 'required disclosure' for the ICLR 2027 conference.

We have **not** used generative AI tools to: help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, implement methods, or to interpret results.

The following were not applicable to this work: generate synthetic data sets, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat dataset, and to support qualitative and thematic data analysis.

Additionally (recommended disclosure), we **used** generative AI tools to: create or edit software code, summarize or analyse existing literature, edit a research paper to improve readability, and to identify relevant literature.

We have reviewed all AI-assisted work. We never directly allowed a generative AI tool to modify our manuscript and instead always iterated for clarity on text or sections and then placed and edited them to fit. All code was checked and developed by at least two authors and AI usage was limited to refactoring or reorganising code and was checked multiple times post-hoc. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

N/A

### REPRODUCIBILITY STATEMENT

In order to ensure reproducibility, we add the pseudocode of the algorithm in Algorithm 1. Furthermore, the Appendix A describes the models, datasets, and hyperparameters required for running our models. Lastly, we make available all code used to generate the experiments at <SEE ATTACHED ZIP>.

## REFERENCES

Martín Abadi, Ashish Agarwal, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

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.

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024.

Babajide O Ayinde, Tamer Inanc, and Jacek M Zurada. Redundant feature pruning for accelerated inference in deep neural networks. *Neural Networks*, 118:148–158, 2019.

Uranik Berisha, Jens Mehnert, and Alexandru Paul Condurache. Variance-based pruning for accelerating and compressing trained networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4973–4982, 2025.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Jerry Chee, Anil Damle, Christopher M De Sa, et al. Model preserving compression for neural networks. *Advances in neural information processing systems*, 35:38060–38074, 2022.

Ting-Wu Chin, Cha Zhang, and Diana Marculescu. Layer-compensated pruning for resource-constrained convolutional neural networks. *ArXiv Preprint arXiv:1810.00518*, 2018.

Jack Choquette, Wishwesh Gandhi, Olivier Giroux, Nick Stam, and Ronny Krashinsky. Nvidia a100 tensor core gpu: Performance and innovation. *IEEE Micro*, 41(2):29–35, 2021.

Xavier Suau Cuadros, Luca Zappella, and Nicholas Apostoloff. Filter distillation for network compression. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3140–3149, 2020.

Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pp. 10323–10337. PMLR, 2023.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pp. 291–326. Chapman and Hall/CRC, 2022.

Felix Goldberg, Yackov Lubarsky, Alexei Gaissinski, Dan Botchan, and Pavel Kisilev. Pruning neural nets by optimal neuron merging. In *International Conference on Pattern Recognition and Artificial Intelligence*, pp. 688–699. Springer, 2022.

Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

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*, pp. 770–778, 2016.

Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4340–4349, 2019.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, 2017.

Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

Norman Jouppi, Cliff Young, Nishant Patil, and David Patterson. Motivation for and evaluation of the first tensor processing unit. *IEEE Micro*, 38(3):10–19, 2018.

Woojeong Kim, Suhyun Kim, Mincheol Park, and Geunseok Jeon. Neuron merging: Compensating for pruned neurons. *Advances in Neural Information Processing Systems*, 33:585–595, 2020.

Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *ArXiv Preprint arXiv:1806.08342*, 2018.

Eldar Kurtić, Elias Frantar, and Dan Alistarh. Ziplm: Inference-aware structured pruning of language models. *Advances in Neural Information Processing Systems*, 36:65597–65617, 2023.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Asriel Levin, Todd Leen, and John Moody. Fast pruning using principal components. *Advances in Neural Information Processing Systems*, 6, 1993.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *ArXiv Preprint arXiv:1608.08710*, 2016.

Jianwei Li, Yijun Dong, and Qi Lei. Greedy output approximation: Towards efficient structured pruning for LLMs without retraining. *ArXiv Preprint arXiv:2407.19126*, 2024.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. *arXiv preprint arXiv:2202.07800*, 2022.

Yajun Liu, Kefeng Fan, Dakui Wu, and Wenju Zhou. Filter pruning by quantifying feature similarity and entropy of feature maps. *Neurocomputing*, 544:126297, 2023.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5058–5066, 2017.

Zelda Mariet and Suvrit Sra. Diversity networks: Neural network compression using determinantal point processes. *ArXiv Preprint arXiv:1511.05077*, 2015.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *ArXiv Preprint arXiv:1611.06440*, 2016.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11264–11272, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient vision transformers with dynamic token sparsification. *Advances in Neural Information Processing Systems*, 34:13937–13949, 2021.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015.

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

Zhuoran Song, Yihong Xu, Zhezhi He, Li Jiang, Naifeng Jing, and Xiaoyao Liang. CP-ViT: Cascade vision transformer pruning via progressive sparsity prediction. *arXiv preprint arXiv:2203.04570*, 2022.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

Lucas Theis, Iryna Korshunova, Alykhan Tejani, and Ferenc Huszár. Faster gaze prediction with dense networks and Fisher pruning. *ArXiv Preprint arXiv:1801.05787*, 2018.

Alexander Theus, Olin Geimer, Friedrich Wicke, Thomas Hofmann, Sotiris Anagnostidis, and Sidak Pal Singh. Towards meta-pruning via optimal transport. *arXiv preprint arXiv:2402.07839*, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Tycho van der Ouderaa, Markus Nagel, Mart Van Baalen, and Tijmen Blankevoort. The llm surgeon. In *International Conference on Learning Representations*, volume 2024, pp. 13245–13255, 2024.

Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. Eigendamage: Structured pruning in the kronecker-factored eigenbasis. In *International conference on machine learning*, pp. 6566–6575. PMLR, 2019.

Zhenyu Wang, Hao Luo, Pichao Wang, Feng Ding, Fan Wang, and Hao Li. VTC-LFC: Vision transformer compression with low-frequency components. *Advances in Neural Information Processing Systems*, 35:13974–13988, 2022.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Changdi Yang, Pu Zhao, Yanyu Li, Wei Niu, Jiexiong Guan, Hao Tang, Minghai Qin, Bin Ren, Xue Lin, and Yanzhi Wang. Pruning parameterization with bi-level optimization for efficient semantic segmentation on the edge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15402–15412, 2023a.

Huanrui Yang, Hongxu Yin, Maying Shen, Pavlo Molchanov, Hai Li, and Jan Kautz. Global vision transformer pruning with hessian-aware saliency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18547–18557, 2023b.

Lu Yu and Wei Xiang. X-pruner: Explainable pruning for vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24355–24363, 2023.

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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9194–9203, 2018.

Shixing Yu, Tianlong Chen, Jiayi Shen, Huan Yuan, Jianchao Tan, Sen Yang, Ji Liu, and Zhangyang Wang. Unified visual transformer compression. *arXiv preprint arXiv:2203.08243*, 2022.

Haonan Zhang, Longjun Liu, Hengyi Zhou, Liang Si, Hongbin Sun, and Nanning Zheng. Fchp: Exploring the discriminative feature and feature correlation of feature maps for hierarchical DNN pruning and compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 32 (10):6807–6820, 2022a.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022b.

Chuanyang Zheng, Kai Zhang, Zhi Yang, Wenming Tan, Jun Xiao, Ye Ren, Shiliang Pu, et al. Savit: Structure-aware vision transformer pruning via collaborative optimization. *Advances in Neural Information Processing Systems*, 35:9010–9023, 2022.

# APPENDIX

## A EXPERIMENTAL DETAILS

**Models** For our experiments on VGG-16 and ResNet-50, we prune the pretrained networks from PyTorch (Paszke et al., 2019) (torchvision weights versions, e.g. `IMAGENET1K_V1`), trained on the ILSVRC 2012 (ImageNet) dataset. The pretrained OPT-125M and LLaMA-2 7B models were loaded from Hugging Face (Wolf et al., 2019).

**VGG-16 baselines** Several comparison methods are re-implemented, including summed absolute weights (SAW) (Li et al., 2016) and PFA-EN (Cuadros et al., 2020). SAW does not provides guidance on the global ranking of units, instead assuming that a practitioner might prune all layers uniformly by the same amount. We therefore use a uniform pruning ratio across layers when implementing these methods. In contrast, PFA-EN performs PCA to decide on a global ranking of units on top of their local importance structure. We apply PFA-EN at the input units rather than the output activations, as we found this to give the best performance. We re-implemented these baselines due to unavailable code or outdated packages. Code to reproduce all experiments is provided in the supplementary material.

**Hyperparameters** Table 3 lists the hyperparameters for all experiments. No fine-tuning is performed after pruning. For the vision models, all pruning is done in a single shot ($T = 1$). For the language models, we use the calibration data and target sizes of LLM Surgeon (van der Ouderaa et al., 2024), where the target size $\alpha$ is the fraction of parameters that remains after pruning.

Table 3: Hyperparameters for all experiments. For the language models, the damping follows the LLM Surgeon defaults (van der Ouderaa et al., 2024), with $\lambda_C$ and $\lambda_H$ added to $\mathbf{C}_l$ and $\bar{\mathbf{H}}_l$ relative to their diagonals.

| | VGG-16 | ResNet-50 | OPT-125M | LLaMA-2 7B |
|---|---|---|---|---|
| Pretrained weights | torchvision | torchvision | Hugging Face | Hugging Face |
| Calibration data | ImageNet train | ImageNet train | WikiText-2 train | WikiText-2 train |
| Calibration samples | Images | Images | 128 sequences | 128 sequences |
| Sequence length | – | – | 2048 tokens | 2048 tokens |
| Evaluation data | ImageNet val | ImageNet val | WikiText-2 test | WikiText-2 test |
| Shots $T$ | 1 | 1 | up to 40 | up to 40 |
| Pruning target | - | - | $\alpha \in \{0.9, \ldots, 0.5\}$ | $\alpha \in \{0.9, \ldots, 0.5\}$ |
| Damping $\lambda_C$, $\lambda_H$ | 0, – | 0, – | 0.01, 0.01 | 0.01, 0.01 |
| Fine-tuning | none | none | none | none |

## B EXTENDED OPT-125M RESULTS

Table 4: Perplexity on WikiText-2 of OPT-125M (dense: 27.65) after structured pruning of rows and columns with SNP and LLM Surgeon (LS), for all numbers of shots and sparsities, extending Table 2. Lower is better; the better method in each pair is in bold.

| | 10% | | 20% | | 30% | | 40% | | 50% | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Shots** | SNP | LS | SNP | LS | SNP | LS | SNP | LS | SNP | LS |
| 1 | **28.48** | 29.30 | **34.95** | 221.06 | **55.13** | 3357 | **132.01** | 3875 | **579.72** | 4258 |
| 2 | **27.93** | 28.70 | **30.01** | 32.01 | **34.89** | 44.59 | **45.13** | 270.25 | **66.19** | 2699 |
| 3 | **27.87** | 30.15 | **29.08** | 31.33 | **34.14** | 34.97 | **43.21** | 50.92 | **63.08** | 106.89 |
| 4 | **27.98** | 28.31 | **29.64** | 30.51 | **34.19** | 37.13 | **41.40** | 45.07 | **56.47** | 69.32 |
| 5 | **27.85** | 28.97 | **29.07** | 31.50 | **32.41** | 33.54 | **41.06** | 45.05 | **54.21** | 59.79 |
| 10 | **27.94** | 28.34 | **28.84** | 30.19 | **32.37** | 33.57 | **39.57** | 43.21 | **52.91** | 55.32 |
| 15 | **27.86** | 28.22 | **28.77** | 31.05 | **32.23** | 33.51 | **38.68** | 42.09 | **49.72** | 57.33 |
| 20 | **27.92** | 28.00 | **28.80** | 29.41 | **32.11** | 33.16 | **38.22** | 39.50 | **49.98** | 50.41 |
| 25 | **27.81** | 28.01 | **28.65** | 30.52 | **31.82** | 33.67 | **38.32** | 40.62 | **49.82** | 51.54 |
| 30 | **27.91** | 27.99 | **28.73** | 29.10 | **31.90** | 32.62 | **38.32** | 41.09 | **50.60** | 50.88 |
| 35 | 28.13 | **28.03** | **28.61** | 28.81 | **31.86** | 38.54 | **37.49** | 37.77 | **48.53** | 50.17 |
| 40 | **27.87** | 28.10 | **28.62** | 28.71 | **31.74** | 31.88 | **37.73** | 38.67 | **48.89** | 49.72 |

## C Comparison with reported LLM baselines

Table 5 compares SNP with results reported for SliceGPT (Ashkboos et al., 2024), for K-OBD (LeCun et al., 1989; Wang et al., 2019) and LLM Surgeon as reported by van der Ouderaa et al. (2024). K-OBD uses the same Kronecker-factored curvature as LLM Surgeon, but scores units independently and does not compensate the remaining weights. Both are reported from a single run with a linear 40-shot schedule to 50% sparsity, such that lower sparsities are reached after proportionally fewer shots (8 shots for 10%, 16 for 20%, and so on). SliceGPT is a single-shot method that removes embedding dimensions rather than units, such that its slicing fraction does not correspond to the fraction of parameters removed; we report its results with 128 calibration sequences, and its evaluation gives a different dense perplexity for LLaMA-2 7B.

Table 5: Perplexity on WikiText-2 of SNP with 5 and 40 shots, compared with results reported for SliceGPT (Ashkboos et al., 2024), and for K-OBD and LLM Surgeon (LS) by van der Ouderaa et al. (2024), on OPT-125M (top) and LLaMA-2 7B (bottom). Lower is better; the best value in each row is in bold, and – marks settings not evaluated. †Values in italics are reported by other works. Numbers in parentheses give the number of shots used by each reported method: SliceGPT is single-shot, and K-OBD and LS are reported from a single run with a linear 40-shot schedule to 50% sparsity, reaching lower sparsities after fewer shots. Only at 50% sparsity do the reported K-OBD and LS results use the same number of shots as SNP with 40 shots; all other comparisons are indicative only.

| **OPT-125M** | | | | | |
|---|---|---|---|---|---|
| | SNP | | | | |
| **Sparsity** | 5 shots | 40 shots | SliceGPT† | K-OBD† | LS† |
| Dense | 27.65 | | *27.64* | *27.65* | *27.65* |
| 10% | **27.85** | 27.87 | *29.33* (1) | *27.97* (8) | *28.01* (8) |
| 20% | 29.07 | **28.62** | *34.53* (1) | *29.89* (16) | *28.73* (16) |
| 30% | 32.41 | **31.74** | *44.61* (1) | *36.54* (24) | *31.82* (24) |
| 40% | 41.06 | **37.73** | – | *47.54* (32) | *38.47* (32) |
| 50% | 54.21 | **48.89** | – | *75.95* (40) | *49.78* (40) |

| **LLaMA-2 7B** | | | | | |
|---|---|---|---|---|---|
| | SNP | | | | |
| **Sparsity** | 5 shots | 40 shots | SliceGPT† | K-OBD† | LS† |
| Dense | 5.12 | | *5.47* | *5.12* | *5.12* |
| 10% | 5.49 | – | *5.96* (1) | *5.48* (8) | ***5.25*** (8) |
| 30% | 8.27 | – | *8.64* (1) | *15.43* (24) | ***7.83*** (24) |
| 50% | 19.00 | **14.88** | – | *46.64* (40) | *15.38* (40) |

These comparisons differ in how directly they can be interpreted. At 50% sparsity, SNP with 40 shots and the reported K-OBD and LLM Surgeon results use the same number of shots, and our pipeline is built on the LLM Surgeon codebase with matched settings; our own 40-shot runs of LLM Surgeon reproduce the reported values closely (Section 4.3). These results are therefore directly comparable. At lower sparsities, SNP uses either fewer (5) or more (40) shots than the reported runs, and the comparison with SliceGPT involves a different pruning structure and evaluation; these comparisons are indicative only.

In the directly comparable setting, SNP achieves the lowest perplexity on both models: 48.89 against 49.78 for LLM Surgeon on OPT-125M, and 14.88 against 15.38 on LLaMA-2 7B, with K-OBD far behind on both. On OPT-125M, SNP with 40 shots also achieves the lowest perplexity at every lower sparsity, and SNP with only 5 shots outperforms K-OBD and SliceGPT at all reported sparsities. On LLaMA-2 7B, LLM Surgeon with 8 and 24 shots achieves lower perplexity than SNP with 5 shots at 10% and 30% sparsity, respectively (5.25 against 5.49, and 7.83 against 8.27). With 5 shots, SNP matches K-OBD at 10% sparsity (5.49 against 5.48), outperforms it substantially at 30% (8.27 against 15.43), and outperforms SliceGPT at both.

### C.1 OPT-125M Predicted vs True Loss Impacts of Pruning

Figure 5 and 6 extends Figure 4 to all 12 blocks of OPT-125M. For each linear layer, units are pruned in a single shot, and we compare the true change in loss with the change predicted by SNP-Fisher and by LLM Surgeon.

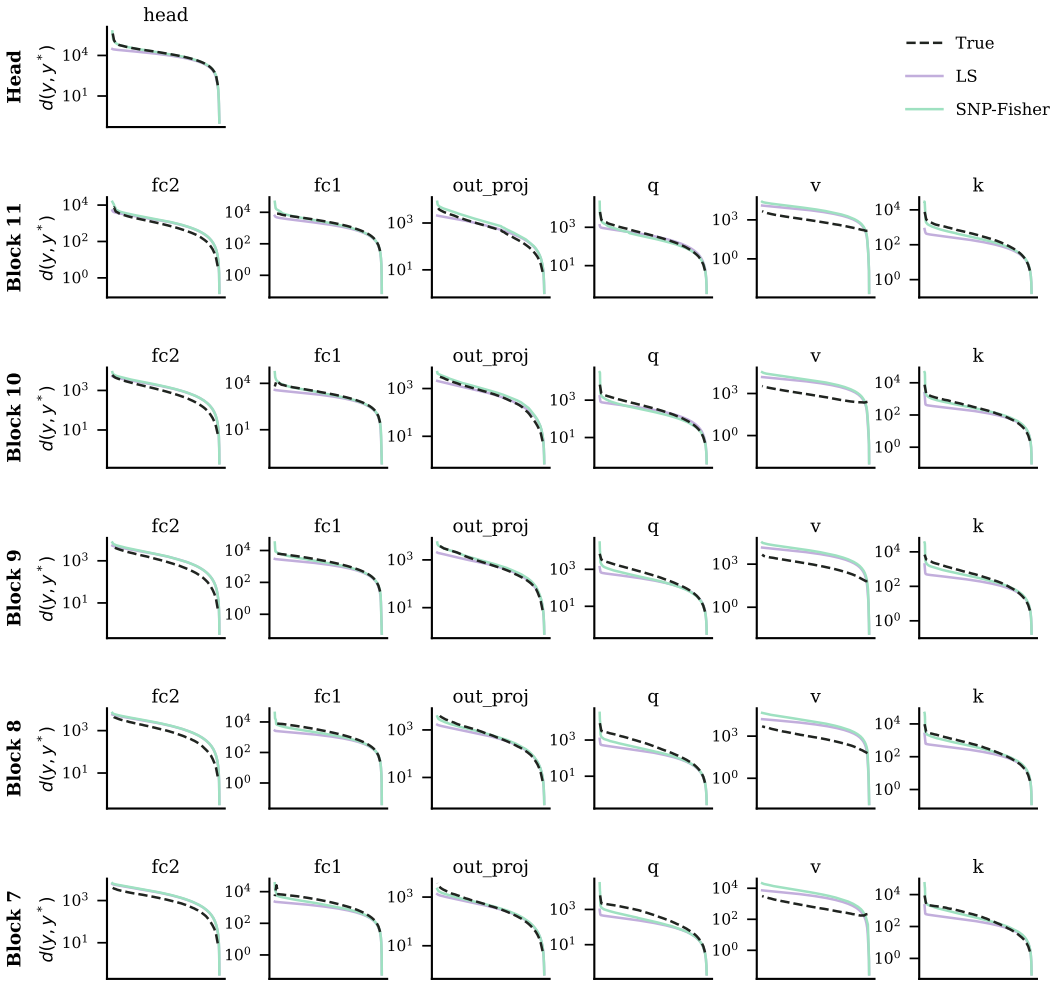



Figure 5: **Predicted and true loss change when pruning units of all blocks of OPT-125M.** Rows correspond to blocks 7-12 and columns to the linear layers of each block (fc1, fc2, out_proj, and the Q, K and V projections). Units are pruned from right to left in a single shot; curves show the true change in loss and the change predicted by SNP-Fisher and by LLM Surgeon. As in the main text, the SNP-Fisher prediction stays close to the true loss as more units are pruned, whereas LLM Surgeon's prediction diverges.

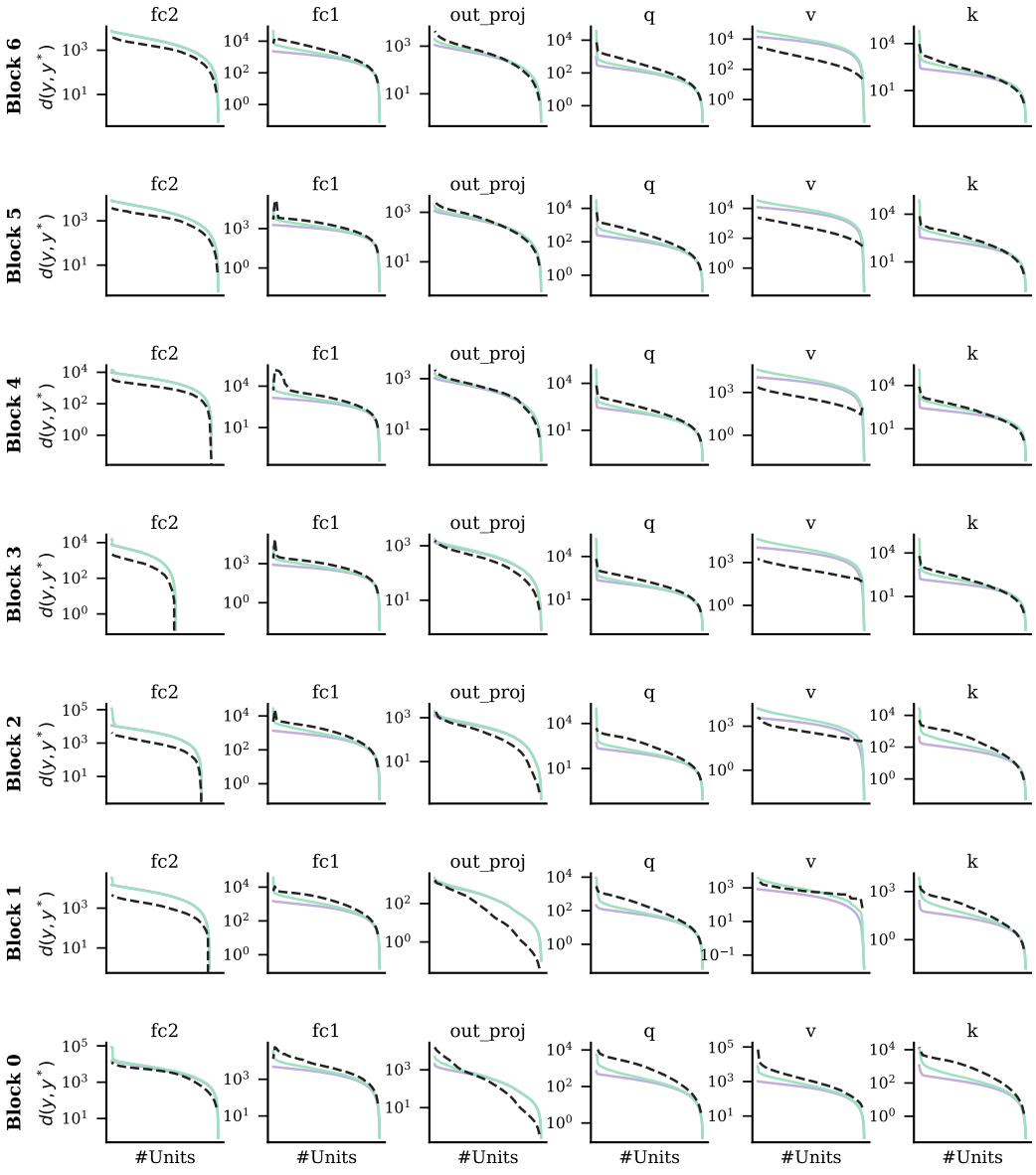



Figure 6: **Predicted and true loss change when pruning units of all blocks of OPT-125M.** Rows correspond to blocks 1-6 and columns to the linear layers of each block (fc1, fc2, out_proj, and the Q, K and V projections). Units are pruned from right to left in a single shot; curves show the true change in loss and the change predicted by SNP-Fisher and by LLM Surgeon. As in the main text, the SNP-Fisher prediction stays close to the true loss as more units are pruned, whereas LLM Surgeon's prediction diverges.