AN ERROR DECOMPOSITION FOR MERGING AND PRUNING NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

While pruning removes units from a trained network, and merging combines several trained networks into one, many methods from both fields can be expressed as linear layer-wise compression methods. We introduce an error decomposition for such methods, which splits the layer-wise compression error exactly into a reconstruction error, which arises from feature information lost through compression, and a commutation error, which arises from the method's failure to commute with the network nonlinearities. Propagating these two local errors through the neural network gives a calibration-data based estimate of the output deviation between the compressed and initial network(s), which can be used to rank methods. Further the analysis reveals that the decoder—the linear operation that reconstructs the original features from the compressed representation—can be fitted independently in closed form once the encoder is fixed, because only the encoder is constrained to commute with the nonlinearity. Paired with the fitted decoder, a simple neuron-importance selection encoder, which has no commutation error, outperforms the considered merging baselines, including ZipIt and PLeaS, on same-task benchmarks across MLP, convolutional, and residual architectures. It performs on par with selecting the encoder using the predicted output deviation, showing that the decoder accounts for most of the downstream improvement. For Llama models, our experiments show that the derived reconstruction and calibration-selected axis allocation changes improve over FLAT-LLM constructions at matched parameter counts.

1 INTRODUCTION

Two machine learning literatures compress neural networks after training, without any gradient updates. *Pruning* reduces the size of a single model by removing neurons, channels, or ranks of its weight matrices (LeCun et al., 1989; Li et al., 2017; Frantar & Alistarh, 2022). *Merging* combines two or more models into one model of the original size by aligning and averaging neurons across models (Wortsman et al., 2022; Ainsworth et al., 2023; Stoica et al., 2024). Both have been unified in the framework of *generalized pruning* (Morelli & Eckstein, 2026): a large model is reduced by layer-wise linear maps, where in the case of merging the large model is the ensemble of all models, run in parallel with shared input and output. At every layer an *encoder* kernel compresses the feature space and a *decoder* kernel maps it back. In this view many popular merging and pruning methods correspond to different choices of these kernels (see Appendix E).

In this paper, we introduce a layer-wise error decomposition for generalized pruning methods that splits the difference of activations between the large and compressed models at every layer into reconstruction and commutation errors, together with the transported upstream error. This decomposition reveals that the decoder can be independently fitted, and matching it with a zero commutation error selection encoder, which prioritizes features by their downstream effect, yields improved performance over ZipIt (Stoica et al., 2024) and PLeaS (Nasery et al., 2025) baselines on MLP, convolutional, and residual architectures. The local error terms of the decomposition can be propagated through the network, allowing us to predict the performance of different methods and select the optimal encoder. By analyzing and adapting FLAT-LLM (Tian et al., 2026), we also show that we can improve pruning performance on Llama models (Touvron et al., 2023; Grattafiori et al., 2024).

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.

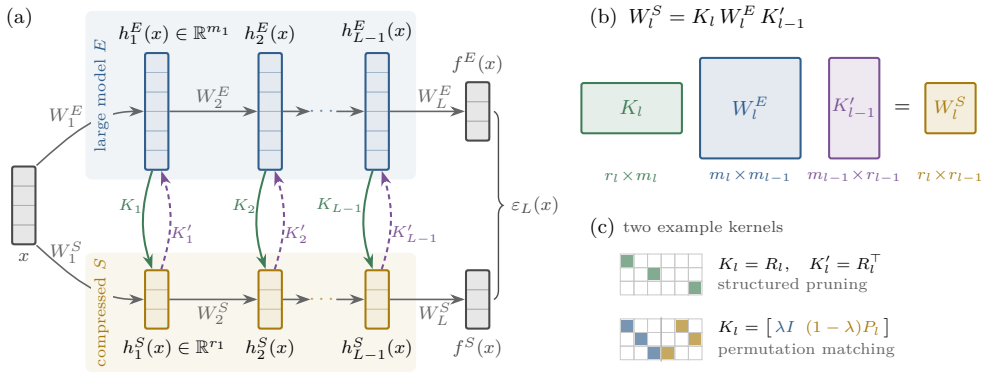


Figure 1: Generalized pruning. (a) The large model E (blue) is compressed into the smaller model S (yellow) using layer-wise encoders (K_l) and decoders (K'_l); $\varepsilon_L(x)$ is the difference in outputs of the two models. (b) These encoders and decoders are directly applied to the weights of each layer, resulting in smaller weight matrices for the model S. (c) Both pruning and merging methods like structured pruning and permutation matching can be expressed through the generalized pruning framework.

2 Preliminaries

Before introducing the error decomposition characterizing different pruning and merging methods, we present the *generalized pruning* framework introduced by Morelli & Eckstein (2026). For simplicity we show the merging case for two models. The construction extends to an arbitrary number of models by forming the ensemble of all of them. Networks are written without biases for readability, and the final layer is linear.

Let A and B be L-layer networks with weight matrices $W_l^M \in \mathbb{R}^{n_l \times n_{l-1}}$, activations $h_0^M(x) := x$, $h_l^M(x) := \sigma_l(W_l^M h_{l-1}^M(x))$, and outputs $f^M(x) := W_L^M h_{L-1}^M(x)$ for $M \in \{A, B\}$. Let E be the L-layer network that we want to compress into a smaller model. For pruning, E is simply the large model itself and for merging, E is the ensemble that runs A and B in parallel on a shared input and averages their outputs:

$$W_l^E := \begin{bmatrix} W_l^A & 0 \\ 0 & W_l^B \end{bmatrix} \text{ for } l \in \{1, \dots, L-1\}, \qquad W_L^E := \begin{bmatrix} \lambda W_L^A & (1-\lambda) W_L^B \end{bmatrix}, \tag{1}$$

For this merging ensemble, $h_0^E(x) := \left[\begin{smallmatrix} x \\ x \end{smallmatrix}\right]$ and $h_l^E(x) := \sigma_l(W_l^E h_{l-1}^E(x))$, so that $f^E(x) = W_L^E h_{L-1}^E(x) = \lambda f^A(x) + (1-\lambda) f^B(x)$. For pruning, instead, E is the single large model and $h_0^E(x) := x$. We write m_l for the width of layer l of E, i.e. $m_l = 2n_l$ for merging and $m_l = n_l$ for pruning.

To reduce the size of E, layer-wise linear transformations are applied to the weights. For every hidden layer $l \in \{1, \dots, L-1\}$, we fix a target width $r_l \le m_l$, an *encoder kernel* $K_l \in \mathbb{R}^{r_l \times m_l}$, and a *decoder kernel* $K'_l \in \mathbb{R}^{m_l \times r_l}$. Input and output are not compressed, which we encode by the boundary conventions $K_L := I_{n_L}$ and $K'_0 := \left[\begin{smallmatrix} I_{n_0} \\ I_{n_0} \end{smallmatrix}\right]$ for merging (both models read the same input), respectively $K'_0 := I_{n_0}$ for pruning. The compressed model is

$$W_l^S := K_l W_l^E K'_{l-1}, \qquad h_0^S(x) := x, \quad h_l^S(x) := \sigma_l(W_l^S h_{l-1}^S(x)), \quad f^S(x) := W_L^S h_{L-1}^S(x). \tag{2}$$

Popular merging and pruning methods correspond to specific kernel choices. Permutation-based merging (Entezari et al., 2022; Ainsworth et al., 2023) uses $K_l = [\lambda I \quad (1-\lambda) P_l]$, $K'_l = \left[\begin{smallmatrix} I \\ P_l^\top \end{smallmatrix}\right]$ with P_l a permutation matrix. Together with the boundary conventions this yields $W_l^S = \lambda W_l^A + (1-\lambda)\, P_l W_l^B P_{l-1}^\top$ for all l (reading $P_0 = P_L = I$), which is exactly the λ-interpolation of aligned weights of Ainsworth et al. (2023), and the compressed features track the aligned convex combination $K_l h_l^E(x) = \lambda h_l^A(x) + (1-\lambda) P_l h_l^B(x)$. Structured pruning corresponds to the selection pair

$K_l = R_l$, $K'_l = R_l^\top$, where the rows of $R_l \in \{0,1\}^{r_l \times m_l}$ are r_l distinct rows of I_{m_l}, so that $K'_l K_l$ retains the selected features and deletes the rest. For an extensive list of methods expressed in the generalized pruning framework we refer to Appendix E.

The main text treats fully connected feedforward networks with coordinatewise activations. Convolutions (kernels acting on channels), pooling, normalization layers, residual streams, and attention require additional conventions, which we describe in Appendix G.

3 Error Decomposition

Generalized pruning methods aim to minimize the gap between the large and the compressed model as measured by a downstream metric such as accuracy or perplexity. These metrics are nonlinear in the model outputs, and accuracy is additionally discontinuous. For a fixed input x we therefore study the *output deviation*, the difference of the outputs of the original and the compressed model

$$\varepsilon_L(x) := f^E(x) - f^S(x), \tag{3}$$

and use averages over the data distribution when estimating.

Throughout, $\|\cdot\|$ is the euclidean and $\|\cdot\|_2$ the spectral norm, $z_l^E(x) := W_l^E h_{l-1}^E(x)$ and $z_l^S(x) := W_l^S h_{l-1}^S(x)$ are the pre-activations of E and S. The object we track through the network is the layer-wise pre-activation deviation

$$\varepsilon_l(x) := K_l z_l^E(x) - z_l^S(x), \qquad l \in \{1, \dots, L\} : \tag{4}$$

$K_l z_l^E(x)$ is the compressed representation obtained by applying the encoder directly to the activations of E, and $z_l^S(x)$ is what S actually computes. At the first layer, $\varepsilon_1(x) = K_1 W_1^E(h_0^E(x) - K'_0 x) = 0$ given by the boundary convention, as nothing has been compressed before the first pre-activation. For the final linear layer we use $K_L = I$, so $\varepsilon_L(x) = W_L^E h_{L-1}^E(x) - W_L^S h_{L-1}^S(x)$, which is exactly equivalent to equation 3. Thus the pre-activation deviation starts at zero for the input layer and ends at the output deviation. In the following we study how it arises and propagates through the network. Two mechanisms lead to a deviation between the models at each layer: when compressing the weight matrices and mapping them back to the larger model with $K'_l K_l$, part of the features are forgotten, and the encoder fails to commute with the nonlinearity. These local layer-wise errors are then propagated through the model. We first isolate the two local mechanisms by assuming there is no pre-activation deviation at the prior layer.

Lemma 1. For $l \in \{1, \dots, L-1\}$, assume $\varepsilon_l(x) = 0$. Then

$$\varepsilon_{l+1}(x) = K_{l+1} W_{l+1}^E \big(I - K'_l K_l\big) h_l^E(x) \;+\; W_{l+1}^S \big(K_l \sigma_l(z_l^E(x)) - \sigma_l(K_l z_l^E(x))\big). \tag{5}$$

The proof is given in Appendix B.1.

The first term is the **reconstruction error**,

$$\varepsilon_{l+1}^{\mathrm{rec}}(x) := K_{l+1} W_{l+1}^E (I - K'_l K_l) h_l^E(x).$$

It is the error caused by losing feature information when compressing to a lower-dimensional representation.
The second term is the **commutation error**,

$$\varepsilon_{l+1}^{\mathrm{com}}(x) := W_{l+1}^S \big(K_l \sigma_l(z_l^E(x)) - \sigma_l(K_l z_l^E(x))\big).$$

It is the difference between encoding before the nonlinearity and encoding afterward.

Without the assumption of having a deviation of zero at previous layers the same pre-activation deviation acquires a third term, carrying the previous layer's error forward.

Theorem 2. For every $l \in \{1, \dots, L-1\}$,

$$\varepsilon_{l+1}(x) = \varepsilon_{l+1}^{\mathrm{rec}}(x) + \varepsilon_{l+1}^{\mathrm{com}}(x) + \varepsilon_{l+1}^{\mathrm{prop}}(x), \tag{6}$$

$$\varepsilon_{l+1}^{\mathrm{prop}}(x) := W_{l+1}^S \Big(\sigma_l\big(K_l z_l^E(x)\big) - \sigma_l\big(K_l z_l^E(x) - \varepsilon_l(x)\big)\Big). \tag{7}$$

If σ_l is L_{σ_l}-Lipschitz, then

$$\|\varepsilon_{l+1}(x)\| \le \|\varepsilon_{l+1}^{\mathrm{rec}}(x)\| + \|\varepsilon_{l+1}^{\mathrm{com}}(x)\| + L_{\sigma_l} \|W_{l+1}^S\|_2 \, \|\varepsilon_l(x)\|. \tag{8}$$

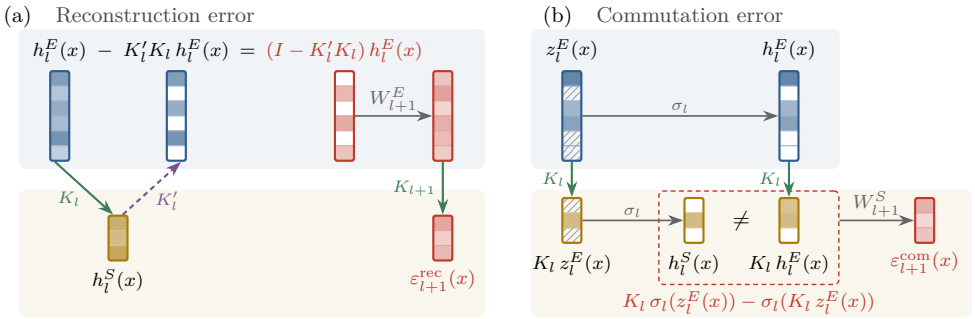


Figure 2: The two local errors (red) at a single layer of E (blue) and S (yellow). (a) The reconstruction error for a selection kernel, which removes neurons and sets their activation to zero (white). The difference to the reconstructed activations is pushed through the next weights and encoder to give the error in the next pre-activations. (b) The commutation error for an averaging kernel with ReLU, which sets negative values (grey) to zero (white). Averaging neurons can change the sign of the pre-activations.

The proof is given in Appendix B.2.

Unrolling the recursion from $\varepsilon_1(x) = 0$ bounds the output deviation by the norms of the local error terms alone.

Theorem 3. Let each σ_l, $l \in \{1, \dots, L-1\}$, be L_{σ_l}-Lipschitz. Then, pointwise in x,

$$\|\varepsilon_L(x)\| \le \sum_{l=2}^{L} \Big(\prod_{j=l}^{L-1} L_{\sigma_j} \|W_{j+1}^S\|_2 \Big) \Big(\|\varepsilon_l^{\mathrm{rec}}(x)\| + \|\varepsilon_l^{\mathrm{com}}(x)\| \Big), \tag{9}$$

with the empty product equal to 1.

The proof is given in Appendix B.3.

4 Minimizing the Local Errors

The local reconstruction and commutation errors help to explain the differences among merging and pruning methods. Flexible encoder-decoder pairs can preserve more feature information, whereas restricted encoders are more likely to commute with the nonlinearity. To understand this trade-off better we study the two errors separately and characterize the kernel choices that minimize each.

We begin by analyzing the reconstruction error. We initially disregard how the lost feature information affects the downstream model and focus on how well the activations of E can be recovered from a representation of dimension r_l:

$$\min_{K_l', K_l} \ \mathbb{E}\big\|(I - K_l' K_l)\, h_l^E(x)\big\|^2 \qquad \text{over kernel pairs of target width } r_l. \tag{10}$$

This is the principal component problem. Let

$$\Sigma_l := \mathbb{E}\big[h_l^E(x) h_l^E(x)^\top\big]$$

denote the uncentered second moment of the activations. The optimum is attained when $K_l' K_l$ is the orthogonal projection onto the subspace spanned by the leading r_l eigenvectors of Σ_l.

Solving equation 10 discards the importance of different directions for the overall model. We therefore measure the residual in a metric G that prices each direction by its downstream effect, $\|v\|_G^2 := v^\top G v$ for a positive semidefinite G, which turns equation 10 into a generalized principal component problem.

Lemma 4. Let G be positive semidefinite and let $\mu_1 \ge \mu_2 \ge \dots$ be the eigenvalues of the priced second moment $G^{1/2} \Sigma_l\, G^{1/2}$. Then

$$\min_{K_l', K_l} \ \mathbb{E}\big\|(I - K_l' K_l)\, h_l^E(x)\big\|_G^2 \ = \ \sum_{i > r_l} \mu_i, \tag{11}$$

over all kernel pairs of target width r_l, attained by whitening with G, projecting onto the leading eigendirections, and unwhitening. For $G = I$ this recovers equation 10.

The proof, a reduction to the Eckart–Young–Mirsky theorem, is given in Appendix B.4.

For a selection encoder, G weights reconstruction errors by their estimated effect on the output. We use this weighted error to choose which features to keep, giving the *priced selection encoder* used in our experiments. Appendix C.4 derives the corresponding paired and averaged Jacobian pricing and compares them empirically.

If there is no activation function between layers (e.g. in some parts of transformers) kernels can be optimized by minimizing this reconstruction error. In standard feedforward networks, every weight matrix is followed by a (coordinatewise) nonlinear activation function. The chosen encoder then has to commute with the activation. The failure to do so is quantified by the commutation error.

Lemma 5. Let σ act coordinatewise with $\sigma(0) = 0$.

(i) If every row of K has at most one nonzero entry, equal to one (i.e. K selects or permutes features) then $\sigma(Ku) = K\sigma(u)$ for all u, for every such σ.

(ii) If σ is in addition positively homogeneous (e.g. ReLU or leaky ReLU), the same holds for *scaled* selections: rows with at most one nonzero entry, which is nonnegative.

(iii) If some row of K has two or more nonzero entries, then $\sigma(Ku) = K\sigma(u)$ for all u forces σ to be additive, and hence linear for any continuous σ.

The proof is given in Appendix B.5. The ReLU specialization was previously proved by Robin (2019).

Selection kernels (i.e. structured pruning) have zero commutation error, however selection is a very limiting constraint for the reconstruction error. On the other extreme, full linear kernels, which minimize the reconstruction error, usually have a large commutation error. Methods in between (e.g. (permuted) weight averaging, clustering) use kernel classes that are less restrictive than selection and commute with a smaller error than arbitrary linear maps.

Blending two pre-activations a and b with weights $(\lambda, 1-\lambda)$ leaves the gap

$$\big|\lambda\sigma(a) + (1-\lambda)\sigma(b) - \sigma\big(\lambda a + (1-\lambda)b\big)\big| \;\le\; 2\lambda(1-\lambda)\, L_\sigma\, |a-b|, \tag{12}$$

for any L_σ-Lipschitz σ, which vanishes when the two units are equivalent. By Jensen's inequality the gap is nonnegative for convex σ such as ReLU. For ReLU itself the bound tightens by a factor of two to $\lambda(1-\lambda)\,|a-b|$, which is attained (see equation 16). Pairing highly correlated units keeps the averaged pre-activations close and the commutation error small. While the two local errors promote opposite kernel classes they can thus be jointly optimized if the matched units are similar.

4.1 Locally optimal decoder

Many merging and pruning methods couple the encoder and decoder by choosing both from one kernel class, seeking a favorable trade-off between commutation and reconstruction error. Generalized pruning does not require this coupling, and equation 5 shows that only the encoder must commute with the nonlinearity. This motivates methods such as PLeaS (Nasery et al., 2025), which combines a restricted encoder with an unrestricted decoder.

Lemma 6. For every $l \in \{1, \dots, L-1\}$,

$$\varepsilon_{l+1}(x) = K_{l+1} W^E_{l+1}\big(h^E_l(x) - K'_l\, h^S_l(x)\big). \tag{13}$$

The proof is given in Appendix B.7.

This is a reformulation of equation 6: once the encoder and upstream kernels are fixed, optimizing K'_l amounts to choosing the linear map that best reconstructs $h^E_l(x)$ from $h^S_l(x)$. This gives the closed-form decoder below.

Theorem 7. Fix the encoder K_l and all kernels upstream, so that $h^S_l(x)$ is determined, and write $\Sigma^{SS}_l := \mathbb{E}[h^S_l(x)(h^S_l(x))^\top]$, $\Sigma^{ES}_l := \mathbb{E}[h^E_l(x)(h^S_l(x))^\top]$. Then

$$K'^{\star}_l := \Sigma^{ES}_l \big(\Sigma^{SS}_l\big)^\dagger \tag{14}$$

where † denotes the Moore–Penrose pseudoinverse. This decoder minimizes $\mathbb{E}\|M(h_l^E(x) - K' h_l^S(x))\|^2$ for every matrix M simultaneously, and in particular minimizes $\mathbb{E}\|\varepsilon_{l+1}(x)\|^2$ at this layer irrespective of the downstream encoder K_{l+1} and the pricing metric G.

The proof is given in Appendix B.8.

Remark 1. The least-squares decoder equation 14 is independent of M, hence of the metric G of Lemma 4. Thus, for a fixed regressor and an unrestricted decoder, downstream importance cannot improve this local quadratic objective by changing the least-squares decoder. It can still influence the encoder and constrained decoder classes. Regularization can also break the independence of G. Adding $\eta\|K_l'\|_F^2$ changes the normal equations to $G\big(K_l'\Sigma_l^{SS} - \Sigma_l^{ES}\big) + \eta K_l' = 0$. The penalty term is not weighted by G, so the regularized optimum depends on G.

Theorem 7 leaves open which activations should be used to fit the decoder. Using the realized $h_l^S(x)$ means fitting against the features that the compressed model actually produces. These features are available only after S has been built up to layer l, so we call the resulting decoder a *post-build* decoder. Because its fitting inputs include the errors accumulated in earlier layers, it can compensate for those errors. Alternatively, we can replace $h_l^S(x)$ with $\sigma_l(K_l z_l^E(x))$, which can be computed from E alone. The resulting *pre-build* decoder can therefore be fitted before constructing the compressed model. The two decoders coincide under exact tracking and otherwise differ because of the error accumulated in preceding layers. Both are deployable, and Theorem 7 does not determine which yields better downstream performance.

For BatchNorm models we need to be careful when computing the post-build decoder as the activation distribution changes when re-calibrating the running statistics of each BatchNorm layer. After fitting and folding a decoder into the next layer, we re-estimate the running statistics *before* fitting the next decoder. Each decoder is therefore fitted to the activation distribution produced by the already-updated model at inference. Appendix C.1 ablates this synchronization, while Section 6.2 compares pre- and post-build fitting.

REPAIR (Jordan et al., 2023) can be viewed as a restricted diagonal-affine decoder obtained by moment matching. Like a diagonally constrained version of equation 14, REPAIR applies a diagonal-affine correction. REPAIR chooses its coefficients by matching moments rather than by least squares, so the resulting corrections generally differ.

5 Predicting the output deviation

Equation 9 yields an upper bound on the output deviation, which is independent of the activations of the compressed network S. This gives an efficient way to compare merging or pruning methods, by storing the activations of network E from one forward pass and applying the different kernels of the methods to the weights and activations of E. However in practice this bound is too loose and overshoots the actual measured output deviation by orders of magnitude. In the following we introduce an *estimate* of the output deviation that can also be computed from E, a calibration dataset, and the kernels.

For fixed kernels, Theorem 2 gives an exact recursion for the layerwise deviation using the activations of E, without a separate forward pass through an explicitly constructed S. However, this still requires evaluating the nonlinearity for the pre-activations that S would produce, which effectively amounts to a forward pass through S. We thus replace the nonlinear error propagation in equation 6 by its linearization at $K_l z_l^E(x)$, using $D_l := \operatorname{diag}\big(\sigma_l'(K_l z_l^E(x))\big)$. Starting from $\varepsilon_1(x) = 0$, this gives

$$\hat{\varepsilon}_L(x) = \sum_{l=2}^{L} T_l(\varepsilon_l^{\mathrm{rec}}(x) + \varepsilon_l^{\mathrm{com}}(x)), \quad T_l := W_L^S D_{L-1} \cdots W_{l+1}^S D_l, \quad T_L := I, \tag{15}$$

where T_l transports an error created at layer l to the output. Equation 15 prices every layer's kernel choice by its own transported contribution to the output. It is exact whenever the σ_l are linear (Proposition 9), hence in particular at the linear interfaces where dense kernels have no commutation error. For ReLU an estimation error can arise when the sign differs between the encoded pre-activation of E and the actual pre-activations of S. Derivations, the sign flip analysis, and the extension to refitted decoders and residual architectures are collected in Appendix F.

Table 1: Comparison of predicted and measured mean squared output deviations. For each architecture, we compare six methods with their native decoders. For each endpoint pair, column two reports Spearman's rank correlation coefficient $\rho \in [-1, 1]$ between the predicted and measured deviations, and column three reports the median ratio of the predicted to the measured deviation. Columns four and five report the correlations between the measured deviation and the increase in loss and the reduction in accuracy respectively. For BatchNorm architectures, candidate-specific running statistics are re-estimated on the calibration set and incorporated into the predictor (Appendix C.5). All entries are the mean $\pm$ sample standard deviation over three endpoint pairs.

Architecture	pred. rank	pred./meas. ratio	dev.→loss	dev.→acc
MLP (BN)	1.000 ± 0.000	1.01 ± 0.01	0.77 ± 0.10	0.81 ± 0.03
MLP (no BN)	1.000 ± 0.000	0.94 ± 0.02	0.94 ± 0.00	0.92 ± 0.09
VGG11 (BN)	0.79 ± 0.16	0.64 ± 0.01	0.68 ± 0.13	0.81 ± 0.17
VGG11 (no BN)	1.000 ± 0.000	1.05 ± 0.01	0.80 ± 0.03	0.58 ± 0.09
ResNet18 (BN)	0.58 ± 0.13	1.28 ± 0.08	0.98 ± 0.03	0.98 ± 0.03
ResNet18 (Fixup)	1.000 ± 0.000	0.77 ± 0.01	0.70 ± 0.24	0.94 ± 0.10

In our experiments in Section 6.2 we use this estimation to predict which method's encoder should be used for a given model E. On the calibration set we construct seven candidate encoders—the six encoders of Section 6.1 and priced selection from Appendix C.4—insert their pre-build least-squares decoders into the estimator, and select the encoder with the smallest predicted squared output deviation. We then equip the selected encoder with the sequential post-build least-squares decoder of Theorem 7.

6 Experiments

We test the locally optimal decoder, priced selection, and the prediction of the output deviation on typical model merging tasks in the vision domain: MLPs trained on MNIST (LeCun et al., 1998), VGG11 (Simonyan & Zisserman, 2015) trained on CIFAR-10, and ResNet18 (He et al., 2016) trained on CIFAR-100 (Krizhevsky, 2009). For MLP and VGG11 we use models with and without BatchNorm (Ioffe & Szegedy, 2015), and for ResNet18 we use standard BatchNorm models and norm-free Fixup models (Zhang et al., 2019). Per architecture we train three disjoint endpoint pairs and report mean $\pm$ sd over pairs. Unless reproducing a published baseline's larger data protocol, experiments use 1,024 unaugmented samples for encoder statistics and prediction, 2,000 for decoder fitting and BatchNorm re-estimation, evaluation on the full test set, $\lambda = 1/2$, and predictions computed on calibration inputs only, never on the test inputs used for the measured deviations. We report the mean squared output deviation $\frac{1}{N}\sum_{i=1}^{N}\left\|f^E(x_i) - f^S(x_i)\right\|_2^2$, where the norm sums over outputs and N is the number of examples in the calibration or evaluation split. Unless a row is explicitly a native decoder diagnostic, a pre-build ablation, or a faithful published-baseline reproduction, every refitted vision result uses the synchronized post-build procedure above. Each experiment below states the exact method set it evaluates.

6.1 Predicting the error of multiple methods

This experiment evaluates six method families wiht their published kernel constructions: selection pruning, permutation alignment, OT fusion, clustering, ZipIt, and CCA alignment (Li et al., 2017; Ainsworth et al., 2023; Singh & Jaggi, 2020; Stoica et al., 2024; Horoi et al., 2024)[1]. Each consists of its encoder paired with the decoder its construction prescribes in Table 14, width-matched ($r_l = n_l$) and without any refit. We rank the six methods by the propagated prediction equation 15 and compare it with the measured deviation after constructing the compressed models. Table 1 shows the results.

Without normalization, the predicted and measured orderings agree on every MLP, VGG11, and Fixup pair. BatchNorm recalibration changes the fused model after assembly, so a predictor that uses only the source models' stored BatchNorm statistics ranks candidates poorly. Therefore we

[1]PLeaS does not appear in this comparison because it is by construction permutation alignment plus a refitted decoder, so it has no unrefitted form to rank.

Table 2: Decoder-refitted merging (accuracy %, mean ± sample sd over three endpoint pairs). Priced selection uses one fixed encoder, while the prediction-based choice ranks all the candidates of Table 1 plus priced selection. Both use our post-build least-squares decoder. Bold marks the best deployable mean, and baseline protocols are in Appendix C.1.

Architecture	endpoints	ZipIt	PLeaS-Act	priced sel.	pred. choice
MLP (BN)	98.2–98.4	97.55 ± 0.23	95.38 ± 0.29	98.15 ± 0.05	$\mathbf{98.17 \pm 0.11}$
MLP (no BN)	97.6–98.0	93.34 ± 0.69	93.47 ± 0.34	$\mathbf{98.03 \pm 0.06}$	$\mathbf{98.03 \pm 0.06}$
VGG11 (BN)	92.1–92.7	90.13 ± 0.12	90.66 ± 0.33	$\mathbf{90.83 \pm 0.12}$	90.78 ± 0.08
VGG11 (no BN)	90.5–91.1	85.86 ± 0.92	88.99 ± 0.03	$\mathbf{90.07 \pm 0.12}$	89.91 ± 0.15
ResNet18 (BN)	77.5–77.9	67.28 ± 0.25	68.68 ± 0.43	73.67 ± 0.02	$\mathbf{74.20 \pm 0.32}$
ResNet18 (Fixup)	71.6–72.1	8.99 ± 1.67	65.68 ± 0.72	$\mathbf{70.17 \pm 0.19}$	70.06 ± 0.34

require an additional prediction step for BatchNorm models because merging changes the activation distribution entering each BatchNorm layer, making the source models' stored running statistics inaccurate for the fused model. For each candidate, we therefore recalibrate its BatchNorm means and variances on the calibration set, fold the resulting diagonal affine transformations into the error propagator, and then compute the predicted output deviation. This BN-aware procedure uses no labels or gradients, although it does require constructing each candidate sufficiently to recalibrate its statistics. With this correction, the rank correlation is 1.000 ± 0.000 on MLP-BN, 0.79 ± 0.16 on VGG11-BN, and 0.58 ± 0.13 on ResNet18-BN. Across architectures, the measured output deviation correlates with the increase in loss by 0.68–0.98 and with the reduction in accuracy by 0.58–0.98. Appendix C.5 provides the construction and additional diagnostic checks.

This experiment thus supports that the output deviation is a useful metric to track and predict in order to estimate the downstream performance of various methods. For models without normalization layers, the output deviation can be estimated from the initial networks alone, without constructing the compressed network. BatchNorm models instead require candidate-specific recalibration.

6.2 The locally optimal decoder and encoder choice

Theorem 7 leaves open whether a pre- or post-build decoder should be used. Table 7 isolates this choice while keeping the permutation encoder fixed. The post-build fit improves over the pre-build fit on every architecture. We suspect that this is due to the ability of the post-build decoder to compensate for the propagated error. We therefore use post-build decoders by default, but note that PLeaS uses a different decoder-fitting procedure (Appendix C.1). The post-build fit also improves over the native and diagonal decoders in every case. It adds 9.7 accuracy points over diagonal repair on the BatchNorm ResNet18 and 8.8 points on Fixup.

Table 8 applies the post-build decoder to seven encoders. Priced selection is within 0.37 accuracy points of the best refitted encoder on every architecture. This shows that the decoder recovers much of the feature information lost by selection, while selection itself has no commutation error. Table 2 therefore reports priced selection as a fixed encoder in addition to the prediction-based encoder choice. Combined with the post-build decoder, priced selection outperforms both listed baselines on all six architectures and matches or exceeds the prediction-based choice on four. For the pred. choice column we choose the encoder type that has the lowest predicted output deviation for a given model pair. It yields strong performance, however it does not consistently outperform the simple priced selection method. Appendix C.2 shows the same encoder–decoder comparison for single-model pruning over compression ratios from 10% to 90%.

To test whether merging needs features from both models, we keep model A's hidden layers and fit its output layer to the mean output of A and B. On our same-task pairs, this already performs well. Refitting A's hidden layers too changes mean accuracy by at most 0.11 points (Appendix C.1). We also evaluate split tasks, with A trained on even classes and B on odd classes. Here the single-model control fits A's output layer to A's even-class and B's odd-class scores. Priced selection and decoder-refitted ZipIt outperform this control on every pair, both when the task is known and when the model must choose among all ten classes. ZipIt has higher ten-class accuracy than priced selection, and refitting its final decoder raises this accuracy by 9.1–25.4 points over released ZipIt (Appendix C.3).

6.3 Structured Llama pruning: adjustments to FLAT-LLM

FLAT-LLM (Tian et al., 2026) uses a dense kernel on the linear value–output path and selection plus a decoder on the nonlinear feedforward path. These choices coincide with our theoretical findings. We retain these choices and make three adjustments.

1. Query–key compatibility. Independent head-wise PCA is not invariant under every rotary transformation, so reduced-dimensional RoPE can introduce a commutation error. We instead retain the same complete rotary pairs for queries and keys, preserving both invariance and coordinate pairing. Because these pairs preserve nearly all measured logit energy ($\tau_h \in [1.000, 1.001]$), we also retain the original $1/\sqrt{d_h}$ normalization.

2. Reconstruction in the output metric. Our decomposition prices a discarded direction by its output error, while PCA preserves the activation variance. On the value–output path we combine the input second moment with the output-sensitivity Gram G. We thus retain the largest output-weighted directions and jointly refit the output projection by least squares. On the feedforward path, we use Wanda (Sun et al., 2024) to select coordinates (i.e. neurons) and the down projection is refitted by the same principle.

3. Rank allocation by predicted output error. We allocate each layer's fixed value rank across heads according to predicted output error. For Llama-2 models we keep FLAT-LLM's proportional pruning split for attention and feedforward modules. Llama-3 is more sensitive to pruning the attention layers, which motivates multiplying the attention keep ratio and reducing the feedforward ratio to preserve the compressed model size. We choose 1.30 from $\{1.00, 1.15, 1.30, 1.45\}$ using held-out WikiText-2 training loss. Appendix D gives the evaluation protocol, ablations, and full results.

Table 3: FLAT-LLM adjustments at $\approx 0.80\times$ parameters. Reconstructed rows use WikiText-only calibration (mean $\pm$ sample sd, three builds).

Model	controlled change	ppl ↓	avg. acc ↑
Llama-2-7B	FLAT-LLM (released impl.)	6.415	59.44
	ours: reconstruction + QK pair selection	6.341 ± 0.001	60.38 ± 0.14
	+ original QK normalization	6.293 ± 0.002	60.19 ± 0.09
	+ head-wise rank allocation	$\mathbf{6.217 \pm 0.003}$	$\mathbf{61.19 \pm 0.20}$
Llama-3-8B	FLAT-LLM (released impl.)	8.454	60.32
	ours: reconstruction + QK pair selection	8.418 ± 0.026	60.89 ± 0.45
	+ attention-protecting split	8.150 ± 0.020	60.51 ± 0.45
	+ original QK normalization + head allocation	$\mathbf{8.070 \pm 0.021}$	$\mathbf{60.98 \pm 0.32}$
Llama-2-13B	FLAT-LLM (published; cross-harness)	5.550	64.38
	ours: full construction	$\mathbf{5.181 \pm 0.002}$	$\mathbf{64.70 \pm 0.59}$

Results. At 7B and 8B, the final rows improve over released FLAT-LLM by $0.198/1.75$ and $0.384/0.66$ perplexity/accuracy points. Three 13B builds reach $5.181 \pm 0.002/64.70 \pm 0.59$, versus the published cross-harness $5.55/64.38$. The full results and protocol are shown in Appendix D.

7 Conclusion

Generalized pruning views merging and structured pruning as layer-wise model compression. Our decomposition separates reconstruction, commutation and propagation errors in each layer. It gives a priced selection encoder and a locally optimal decoder that is fitted independently in closed form. Selection has zero commutation error, while the decoder recovers much of the feature information lost through selection. Together they outperform the considered same-task merging baselines on all six vision architectures. The propagated errors additionally give a calibration-data-based estimate of the output deviation. Our controlled Llama construction improves over the FLAT-LLM baselines; at 13B it retains the gain against the published cross-harness point. Open questions remain for the merging of transformer architectures and pruning of newer and larger LLM models. A competitive allocator that sets all ranks directly from calibration statistics remains open.

AI DISCLOSURE

In this work, we used generative AI tools to help develop theoretical models, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, provide feedback on research experiments and implement methods. We have not used generative AI tools for generating synthetic datasets, proposing hypotheses, assistance with translation, cleaning and reformatting of datasets, supporting qualitative, and thematic data analysis and interpreting of results.

LLMs (mostly Fable 5) were used to aid in the construction of the error decomposition and in writing down proofs for some of the theorems presented. All proofs were verified by the authors. Additionally, LLMs were used to generate code for the experiments, especially the reproduction of existing methods in the generalized pruning framework and the adaptation of existing transformer pruning methods. During writing LLMs were mainly used to check for any mistakes and discrepancies between the text and the actual code implementations and experiments. We take responsibility for the final content of this work, including claims and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All results produced in this work can be reproduced using the accompanying code release. It includes the experiment drivers, fixed configurations and random seeds, calibration-sampling procedures, evaluation commands, software environment, and raw result files used to produce the reported tables. The pretrained checkpoints and datasets are publicly available and are downloaded by the provided scripts, subject to their original licenses. Our methods do not require additional pretraining or finetuning.

For existing methods, we either use the authors' released implementations directly or faithfully implement their kernel constructions within the generalized-pruning framework (see Appendix C.1). Results taken directly from prior work are explicitly marked as published and cross-harness.

REFERENCES

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git Re-Basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations*, 2023.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations*, 2024.

Xin Dong, Shangyu Chen, and Sinno Jialin Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. In *Advances in Neural Information Processing Systems*, volume 30, pp. 4857–4867, 2017.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2022.

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

Zhiren Gong, Zihao Zeng, Zijie Wang, Tiantong Wang, Chau Yuen, and Wei Yang Bryan Lim. CoCurve: Cross-module co-pruning curvature for training-free structured LLM pruning. *arXiv preprint arXiv:2607.17568*, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Fidel A. Guerrero Peña, Heitor Rapela Medeiros, Thomas Dubail, Masih Aminbeidokhti, Eric Granger, and Marco Pedersoli. Re-basin via implicit Sinkhorn differentiation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, volume 5, pp. 164–171, 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.

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.

Stefan Horoi, Albert Manuel Orozco Camacho, Eugene Belilovsky, and Guy Wolf. Harmony in diversity: Merging neural networks with canonical correlation analysis. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 18815–18832. PMLR, 2024.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456. PMLR, 2015.

Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. REPAIR: REnormalizing permuted activations for interpolation repair. In *International Conference on Learning Representations*, 2023.

Woojeong Kim, Suhyun Kim, Mincheol Park, and Geonseok Jeon. Neuron merging: Compensating for pruned neurons. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

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

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2, pp. 598–605, 1989.

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

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

Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? In *International Conference on Learning Representations*, 2016.

Chi-Heng Lin, Shangqian Gao, James Seale Smith, Abhishek Patel, Shikhar Tuli, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. MoDeGPT: Modular decomposition for large language model compression. In *International Conference on Learning Representations*, 2025.

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.

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

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations*, 2017.

Fabian Morelli and Stephan Eckstein. Partial fusion of neural networks: Efficient tradeoffs between ensembles and weight aggregation. In *Proceedings of the 43rd International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2026.

Anshul Nasery, Jonathan Hayase, Pang Wei Koh, and Sewoong Oh. PLeaS – merging models with permutations and least squares. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 30493–30502, 2025.

David A. R. Robin. Neural network compression. Master 1 internship report, Ecole Normale Superieure, 2019. URL https://www.robindar.com/m1-internship/report.pdf.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Elliot Stein, Christine Evers, and Jonathon Hare. Debugging ReBasin: What limits symmetry-based model merging? In *ICML Workshop on Weight-Space Symmetries*, 2026. URL https://openreview.net/forum?id=zffCxj5BdZ.

George Stoica, Daniel Bolya, Jakob Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. ZipIt! Merging models from different tasks without training. In *International Conference on Learning Representations*, 2024.

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

Lintang Sutawika, Hailey Schoelkopf, Leo Gao, Baber Abbasi, Stella Biderman, Jonathan Tow, ben fattori, Charles Lovering, farzanehnakhaee70, Jason Phang, Anish Thite, Fazz, Aflah, Niklas, Thomas Wang, sdtblck, nopperl, gakada, researcher2, tttyuntian, Julen Etxaniz, James A. Michaelov, Chris, Chessing234, Hanwool Albert Lee, Janna, Leonid Sinev, Khalid, Kiersten Stokes, and Zdeněk Kasner. EleutherAI/lm-evaluation-harness: v0.4.12, 05 2026. URL https://zenodo.org/records/20122284.

Norman Tatro, Pin-Yu Chen, Payel Das, Igor Melnyk, Prasanna Sattigeri, and Rongjie Lai. Optimizing mode connectivity via neuron alignment. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Jiayi Tian, Ryan Solgi, Jinming Lu, Yifan Yang, Hai Li, and Zheng Zhang. FLAT-LLM: Fine-grained low-rank activation space transformation for large language model compression. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 2988–3002. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-eacl.156.

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.

Ziyang Wang, Jiangfeng Xiao, Chuan Xiao, Ruoxiang Li, Rui Mao, and Jianbin Qin. GRASPrune: Global gating for budgeted structured pruning of large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 10719–10736, 2026.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*. PMLR, 2022.

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.

Tao Yu, Yongqi An, Kuan Zhu, Guibo Zhu, Ming Tang, and Jinqiao Wang. Improving generalization in llm structured pruning via function-aware neuron grouping. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 27969–27977, 2026.

Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In *International Conference on Learning Representations*, 2019.

APPENDIX CONTENTS

A	Related Work	p. 13
B	Proofs of the Main Results	p. 14
C	Additional Vision Experiments	p. 15
D	Additional Llama Experiments	p. 21
E	Methods as Generalized Pruning	p. 23
F	Propagation Details	p. 25
G	Architecture Extensions	p. 26

A RELATED WORK

Weight averaging and symmetry. Direct weight averaging works when solutions already occupy a compatible basin. This is typically the case, when fine-tuning many models from one shared base model, as in model soups (Wortsman et al., 2022). Independently trained networks usually require alignment because hidden units are permutation-symmetric (Entezari et al., 2022). Git Re-Basin and neuron alignment align hidden units through permutations (Ainsworth et al., 2023; Tatro et al., 2020), while OT fusion also allows soft alignments based on weight or activation similarity (Singh & Jaggi, 2020).

Feature preservation and post-merge correction. ZipIt relaxes the one-to-one alignment (Stoica et al., 2024), CCA-Merge aligns dense correlated subspaces (Horoi et al., 2024), and Partial Fusion keeps unmatched features (Morelli & Eckstein, 2026). REPAIR corrects activation moments (Jordan et al., 2023), while PLeaS fits a least-squares decoder using a permutation matching encoder (Nasery et al., 2025). Our decomposition separates these axes: what the encoder discards or blends and how the decoder reconstructs it.

Reconstruction, commutation, and propagation. The individual error terms of our decomposition have appeared separately in the literature. In pruning, Robin (2019) uses a feature selector followed by a linear reconstructor, chains these maps across layers, and proves the ReLU special case of our commutation characterization: up to admissible row scaling, a linear map commuting with pointwise ReLU is a feature selector. Neuron Merging similarly decomposes weights and folds a scaling map through ReLU under appropriate conditions (Kim et al., 2020). Layer-wise Optimal Brain Surgeon bounds the final prediction degradation by a linear combination of per-layer reconstruction errors (Dong et al., 2017), while NISP propagates final-layer importance scores backward to guide pruning (Yu et al., 2018). On the merging side, Stein et al. (2026) inserts encoder–decoder pairs at representation boundaries and uses a staircase of runtime and folded merges to separate alignment, deployment, folding, and normalization costs. That analysis explicitly notes that reconstruction and ReLU non-commutation remain combined in one transition. We therefore do not claim novelty for these notions individually. Our contribution is to write and exploit the single exact identity that separates newly created reconstruction and commutation error for a generalized-pruning kernel, recursively transports them through depth, and uses the resulting quantities to fit the decoder, use G to choose which features selection keeps, and estimate a compression's output deviation from calibration data.

Structured pruning and low-rank compression. Classical pruning selects units using magnitude, Taylor, or second-order criteria (LeCun et al., 1989; Hassibi & Stork, 1992; Li et al., 2017; Molchanov et al., 2017). Reconstruction-aware methods minimize the downstream change of the model through pruning (He et al., 2017; Luo et al., 2017; Frantar & Alistarh, 2022). SliceGPT rotates the residual stream of transformers before compression (Ashkboos et al., 2024), while FLAT-LLM and MoDeGPT combine structured and low-rank transforms (Tian et al., 2026; Lin et al., 2025). Recent pruning approaches learn global gates, group neurons by function, or retain cross-module curvature terms (Wang et al., 2026; Yu et al., 2026; Gong et al., 2026). Our framework separates the encoder that chooses what is retained from the decoder that reconstructs the discarded features. It also tracks how the resulting errors affect the output. Appendix E gives the exact mappings, and Appendix G describes the architectural restrictions.

B Proofs of the Main Results

B.1 Local error decomposition

Proof of Lemma 1. The assumption gives $h_l^S(x) = \sigma_l(z_l^S(x)) = \sigma_l(K_l z_l^E(x))$, hence $z_{l+1}^S(x) = K_{l+1} W_{l+1}^E K_l' \, \sigma_l(K_l z_l^E(x))$, while $K_{l+1} z_{l+1}^E(x) = K_{l+1} W_{l+1}^E h_l^E(x)$. Subtracting, inserting $\pm K_{l+1} W_{l+1}^E K_l' K_l h_l^E(x)$, and using $K_l h_l^E(x) = K_l \sigma_l(z_l^E(x))$ yields equation 5. □

B.2 Propagation

Proof of Theorem 2. Now $h_l^S(x) = \sigma_l(K_l z_l^E(x) - \varepsilon_l(x))$, so $\varepsilon_{l+1}(x) = K_{l+1} W_{l+1}^E h_l^E(x) - W_{l+1}^S \, \sigma_l(K_l z_l^E(x) - \varepsilon_l(x))$; adding and subtracting $W_{l+1}^S \sigma_l(K_l z_l^E(x))$ and applying the computation of Lemma 1 to the remaining difference yields equation 6. The bound follows from the triangle inequality and $\|\varepsilon_{l+1}^{\mathrm{prop}}(x)\| \le \|W_{l+1}^S\|_2 \, L_{\sigma_l} \, \|\varepsilon_l(x)\|$. □

B.3 Output deviation bound

Proof of Theorem 3. Induction over equation 8 with $\varepsilon_1(x) = 0$ gives equation 9. □

B.4 Reconstruction floor

Proof of Lemma 4. Write $A := G^{1/2} \Sigma_l^{1/2}$, put $\Pi := K_l' K_l$, and note $\mathbb{E}\|(I - \Pi) h_l^E(x)\|_G^2 = \|G^{1/2}(I - \Pi)\Sigma_l^{1/2}\|_F^2$. For invertible G and Σ_l, the matrix $B := G^{1/2} \Pi \, \Sigma_l^{1/2}$ ranges over all matrices of rank at most r_l as the pair (K_l', K_l) does, so by Eckart–Young–Mirsky $\min_B \|A - B\|_F^2 = \sum_{i > r_l} \sigma_i^2(A)$, and $\sigma_i^2(A)$ are the eigenvalues μ_i of $AA^\top = G^{1/2} \Sigma_l G^{1/2}$. The minimizer $B = U_r U_r^\top A$, with U_r the top-r_l eigenvectors of $AA^\top$, corresponds to $\Pi = G^{-1/2} U_r U_r^\top G^{1/2}$, which factorizes as $K = U_r^\top G^{1/2}$, $K' = G^{-1/2} U_r$ with $KK' = I_{r_l}$. Singular G or Σ_l follow by taking limits of invertible perturbations. □

B.5 Commutation of kernel classes

Proof of Lemma 5. Part (i) is immediate: such a K acts by picking and reordering coordinates, which commutes with any coordinatewise map, and the discarded coordinates contribute $\sigma(0) = 0$. For part (ii), a row $c e_j^\top$ with $c \ge 0$ gives $\sigma(c u_j) = c \, \sigma(u_j)$ by positive homogeneity. For part (iii), let $k^\top$ be a row of K with $k_i, k_j \ne 0$ for some $i \ne j$. Testing vectors supported on $\{i, j\}$ with entries s and t gives $\sigma(k_i s + k_j t) = k_i \sigma(s) + k_j \sigma(t)$ for all s, t. Setting $t = 0$ and $s = 0$ separately, then substituting both identities back, yields $\sigma(a + b) = \sigma(a) + \sigma(b)$ for all $a, b \in \mathbb{R}$. □

B.6 ReLU averaging gap

For a row that averages pre-activations a and b with weights λ and $1 - \lambda$, where $0 \le \lambda \le 1$, the commutation gap under ReLU satisfies

$$0 \;\le\; \lambda\sigma(a) + (1 - \lambda)\sigma(b) - \sigma\big(\lambda a + (1 - \lambda) b\big) \;\le\; \lambda(1 - \lambda)\,|a - b|. \tag{16}$$

The bound follows by checking the cases in which a and b have the same or opposite signs. The upper bound is attained when $a = (1 - \lambda) d$ and $b = -\lambda d$ for $d > 0$.

B.7 Decoder residual identity

Proof of Lemma 6. By equation 2, $z_{l+1}^S(x) = W_{l+1}^S h_l^S(x) = K_{l+1} W_{l+1}^E K_l' h_l^S(x)$, while $K_{l+1} z_{l+1}^E(x) = K_{l+1} W_{l+1}^E h_l^E(x)$. Subtracting gives equation 13. □

B.8 Locally optimal decoder

Proof of Theorem 7. If $v \in \ker(\Sigma_l^{SS})$, then $\mathbb{E}[(v^\top h_l^S(x))^2] = 0$, so $v^\top h_l^S(x) = 0$ almost surely and $\Sigma_l^{ES} v = 0$. Hence $\Sigma_l^{ES} (\Sigma_l^{SS})^\dagger \Sigma_l^{SS} = \Sigma_l^{ES}$, and the choice equation 14 makes the residual

uncorrelated with the regressor, $\mathbb{E}[(h_l^E(x) - K_l'^\star h_l^S(x))(h_l^S(x))^\top] = \Sigma_l^{ES} - K_l'^\star \Sigma_l^{SS} = 0$. For any K', write $h_l^E(x) - K' h_l^S(x) = (h_l^E(x) - K_l'^\star h_l^S(x)) + (K_l'^\star - K')h_l^S(x)$. The cross terms vanish, and hence

$$R(K') = R(K_l'^\star) + (K_l'^\star - K')\Sigma_l^{SS}(K_l'^\star - K')^\top \succeq R(K_l'^\star).$$

Therefore $\mathbb{E}\|M(h_l^E(x) - K' h_l^S(x))\|^2 = \mathrm{tr}(MR(K')M^\top)$ is monotone in the semidefinite order. By Lemma 6, the interface tracking error is this quantity with $M = K_{l+1}W_{l+1}^E$.

If Σ_l^{SS} is singular, let $P_l := \Sigma_l^{SS}(\Sigma_l^{SS})^\dagger$, the orthogonal projector onto $\mathrm{range}(\Sigma_l^{SS})$. The complete family of minimizers is

$$K_l'^\star + A(I - P_l),$$

where A is arbitrary. These decoders agree on $\mathrm{range}(\Sigma_l^{SS})$ and therefore act identically on $h_l^S(x)$ almost surely. The choice in equation 14 corresponds to $A = 0$ and is the minimum-norm solution. □

Remark 2 (Estimation conventions). All expectations are estimated on a calibration set, and Σ_l denotes the *uncentered* second moment (means and biases are absorbed by the constant-feature convention of Appendix E). Σ_l, the Gram equation 18, and the refit moments of Theorem 7 are measured on the tensor the next operation actually consumes: post-activation features $h_l^E(x)$, attention-weighted value features at the output-projection input, or realized compressed features for a post-build refit. The decoder reconstructs these features after the nonlinearity, so pre-activation statistics optimize a different objective. Centering the moments is equivalent only when an affine intercept is fitted.

C Additional Vision Experiments

C.1 Baseline fidelity and post-build fitting

BatchNorm synchronization. Recalibrating the BatchNorm layers changes the activations supplied to every subsequent layer. If BatchNorm is recalibrated only once before all decoders are fitted, later regressions use inputs generated with running statistics that become stale as preceding maps are installed. We instead alternate decoder fitting and BatchNorm recalibration: after installing each fitted non-output map, we recalibrate the model before collecting inputs for the next regression. Table 4 isolates this change on held-out pairs using one fixed priced selection encoder. The synchronized procedure outperforms one-shot recalibration on all nine endpoint pairs.

Table 4: BatchNorm-aware decoder fitting (accuracy %, mean ± sample sd over three held-out endpoint pairs). "One-shot post" recalibrates once before fitting the decoders. "Synchronized post" recalibrates after every fitted non-output map, before collecting inputs for the next regression, and is the default used elsewhere.

Architecture	pre-build	one-shot post	synchronized post
MLP-BN	98.093 ± 0.086	97.580 ± 0.350	$\mathbf{98.173 \pm 0.085}$
VGG11-BN	90.697 ± 0.181	86.133 ± 0.798	$\mathbf{91.043 \pm 0.346}$
ResNet18-BN	71.870 ± 0.551	65.343 ± 1.310	$\mathbf{74.027 \pm 0.401}$

PLeaS. Our baseline pins the authors' repository at commit `55b35ee` and calls its released permutation compiler and activation-matching routine directly. We retain its 100 matching batches, negative-Euclidean cost, Hungarian assignment, released Adam fitting procedure, and final BatchNorm reset.

PLeaS uses a different regression objective from the decoder in Section 4.1. Our pre-build decoder regresses on $\sigma_l(K_l z_l^E(x))$, whereas PLeaS uses $K_l \sigma_l(z_l^E(x))$. These features agree only when the encoder commutes with the nonlinearity. PLeaS also optimizes its regression with Adam. Replacing Adam with the closed-form solution while keeping its matching, data, and BatchNorm protocol fixed improves all six architectures by 0.20 to 1.86 accuracy points. This control remains below our post-build fit on every architecture. Like our pre-build decoder, PLeaS only uses activations of E and updates BatchNorm after fitting all maps.

ZipIt. We likewise pin the released repository at commit `1b6ca40` and call its matcher directly. We retain the returned merge and unmerge matrices and use the released defaults $\alpha = 0.3$ and $\beta = 0.125$. We also retain its correlations computed over the full training loader and its subsequent BatchNorm recalibration. We evaluate a full-depth, fixed-width merge because partial zipping retains endpoint heads and changes the inference capacity. We fix the released defaults instead of tuning on a reported pair. On one Fixup pair, the released float32 covariance became non-finite because it subtracts two large raw moments. We recomputed the same population covariance with a centered float64 accumulator and left the matcher unchanged. Table 5 changes only the decoder to our synchronized post-build fit.

We also apply a REPAIR-style moment correction (Jordan et al., 2023) to ZipIt without changing its encoder or decoder. For each corrected channel, we match the fused mean and standard deviation to those of the endpoint features mapped by ZipIt's encoder. We correct convolution outputs and, on ResNets, residual-block outputs using 2,000 unlabeled training images. On BatchNorm MLP and VGG11, ZipIt's existing BatchNorm reset already supplies the convolutional correction.

Table 5: Controls for the released ZipIt encoder (accuracy %, mean ± sample sd over three pairs). "REPAIR-style" retains ZipIt's decoder and matches channel moments; "synchronized LS" changes only the decoder to our sequential post-build fit. A dash means ZipIt's BatchNorm reset already provides the convolutional correction.

Architecture	Released	REPAIR-style	Synchronized LS
MLP (BN)	97.55 ± 0.23	–	$\mathbf{98.12 \pm 0.09}$
MLP (no BN)	93.34 ± 0.69	93.80 ± 0.58	$\mathbf{96.53 \pm 0.43}$
VGG11 (BN)	90.13 ± 0.12	–	$\mathbf{90.94 \pm 0.22}$
VGG11 (no BN)	85.86 ± 0.92	88.65 ± 0.07	$\mathbf{89.75 \pm 0.07}$
ResNet18 (BN)	67.28 ± 0.25	67.03 ± 0.70	$\mathbf{73.99 \pm 0.26}$
ResNet18 (Fixup)	8.99 ± 1.67	62.55 ± 0.29	$\mathbf{69.86 \pm 0.22}$

On Fixup, the paired run gives 9.26 ± 1.80 before and 62.55 ± 0.29 after the moment correction. Averaging the endpoint Fixup scalars instead of inheriting those of the first endpoint changes the corrected mean only to 62.69 ± 0.48.

Keeping one model. For a same-task control, we keep model A and fit only its final linear map to the mean output of A and B on 2,000 unlabeled calibration examples. We also keep A's neurons at every hidden layer and fit all maps sequentially with the post-build decoder. Table 6 shows that fitting the hidden maps changes mean accuracy by at most 0.11 points. Thus, on these same-task pairs, high accuracy after refitting does not by itself show that features from both models are needed.

Table 6: Keeping model A for same-task merging (test accuracy %, mean ± sample sd over three pairs). Both fits target the mean output of A and B.

Architecture	readout fit	all-layer fit
MLP (BN)	98.22 ± 0.08	98.22 ± 0.11
MLP (no BN)	97.90 ± 0.12	97.89 ± 0.11
VGG11 (BN)	92.42 ± 0.18	92.31 ± 0.30
VGG11 (no BN)	90.98 ± 0.19	90.99 ± 0.19
ResNet18 (BN)	77.47 ± 0.20	77.40 ± 0.14
ResNet18 (Fixup)	71.74 ± 0.25	71.74 ± 0.28

Encoder comparison under a common decoder. Table 8 evaluates every encoder considered by the prediction-based choice in Table 2 with the same decoder-fitting rule: a synchronized post-build least-squares decoder is fitted separately for each encoder. This removes the decoder family and fitting procedure as confounds; differences between rows arise from the encoder and its induced compressed representation.

For the seven encoders in Table 8, we also compare the order predicted with a pre-build decoder proxy with the measured output deviations after post-build decoder fitting. The Spearman correlations are 0.976 ± 0.021 (MLP-BN), 1.000 ± 0.000 (MLP), 0.393 ± 0.156 (VGG11-BN), 0.845 ± 0.055

Table 7: The effect of the decoder choice (fused accuracy %, mean ± sd over three pairs). All columns use the same permutation encoder.

Architecture	native	diagonal	pre-build LS	post-build LS
MLP (BN)	96.62 ± 0.23	97.59 ± 0.25	98.07 ± 0.06	$\mathbf{98.13 \pm 0.09}$
MLP (no BN)	93.43 ± 2.44	94.85 ± 0.91	97.07 ± 0.12	$\mathbf{97.14 \pm 0.17}$
VGG11 (BN)	67.85 ± 1.05	88.55 ± 0.11	88.51 ± 0.11	$\mathbf{90.93 \pm 0.25}$
VGG11 (no BN)	85.70 ± 1.36	88.51 ± 0.22	89.75 ± 0.09	$\mathbf{89.80 \pm 0.04}$
ResNet18 (BN)	4.26 ± 0.53	64.21 ± 0.42	66.85 ± 2.14	$\mathbf{73.87 \pm 0.43}$
ResNet18 (Fixup)	12.76 ± 3.03	61.10 ± 1.72	68.48 ± 0.52	$\mathbf{69.87 \pm 0.17}$

Table 8: Encoder comparison under a common decoder-fitting rule (accuracy %, mean ± sample sd over three endpoint pairs). A synchronized post-build least-squares decoder is fitted separately for every encoder. Every row uses the common 1,024-example encoder-statistics pool and the 2,000-example sequential decoder fit. The ZipIt row uses its released matcher and default hyperparameters within this common data protocol. The oracle test-selects among the seven encoders separately for each endpoint pair.

Encoder	MLP-BN	MLP	VGG-BN	VGG	ResNet-BN	Fixup
Selection	98.10 ± 0.08	97.95 ± 0.07	$\mathbf{90.95 \pm 0.10}$	89.71 ± 0.24	72.92 ± 0.14	69.99 ± 0.21
Priced sel.	98.15 ± 0.05	$\mathbf{98.03 \pm 0.06}$	90.83 ± 0.12	$\mathbf{90.07 \pm 0.12}$	73.67 ± 0.02	70.17 ± 0.19
Permutation	98.13 ± 0.09	97.14 ± 0.17	90.93 ± 0.25	89.80 ± 0.04	73.87 ± 0.43	69.87 ± 0.17
OT	98.09 ± 0.08	97.08 ± 0.07	90.67 ± 0.34	89.66 ± 0.12	73.19 ± 0.29	69.69 ± 0.36
Clustering	$\mathbf{98.17 \pm 0.11}$	97.84 ± 0.10	90.92 ± 0.15	88.96 ± 0.19	73.50 ± 0.36	69.53 ± 0.18
CCA	97.91 ± 0.01	95.95 ± 0.33	90.78 ± 0.08	89.87 ± 0.08	$\mathbf{74.04 \pm 0.57}$	$\mathbf{70.23 \pm 0.34}$
ZipIt	98.08 ± 0.10	96.81 ± 0.27	90.87 ± 0.20	89.71 ± 0.13	74.01 ± 0.14	69.79 ± 0.13
Oracle (test-selected)	98.22 ± 0.07	98.03 ± 0.06	91.10 ± 0.11	90.07 ± 0.12	74.21 ± 0.32	70.38 ± 0.22

(VGG11), 0.060 ± 0.561 (ResNet18-BN), and 0.238 ± 0.251 (Fixup), reported as mean ± sample sd over three pairs. Post-build fitting brings the measured deviations of the encoders closer together, but the prediction does not reliably order the remaining differences on VGG11-BN or the residual networks.

C.2 Single-model structured pruning across compression ratios

We apply the same encoder and decoder choices to structured pruning of one model. For each architecture we reuse the first endpoint from each of the three disjoint merge pairs and prune every hidden layer from 10% to 90% in ten-point increments. The MLP and VGG11 have no BatchNorm and ResNet18 uses Fixup, so normalization recalibration does not confound the comparison. Encoder statistics use 1,024 calibration examples, every post-build decoder uses 2,000, and accuracy is measured on the full test set.

We compare selection, priced selection, PCA-leverage selection, dense PCA, average-linkage clustering, Ward clustering, and a single-model adaptation of ZipIt's correlation-merging rule. Standard filter-magnitude pruning is included as a separate baseline. The figure shows two decoder settings. In the top row, each method uses its native decoder. In the bottom row, each encoder receives its own sequential post-build least-squares decoder. The bottom row also shows prediction-based choice: for each model and pruning ratio, the pre-build predictor selects among the seven encoders, and the selected encoder is then fitted with a post-build decoder. Magnitude pruning is plotted in both rows but is not a selection candidate.

Figure 3 shows that the decoder accounts for a large part of the achievable accuracy. At 50% unit pruning, fitting the decoder changes priced selection from 94.42% to 97.16% on the MLP, from 49.08% to 86.48% on VGG11, and from 1.00% to 64.95% on ResNet18. The corresponding magnitude-pruning results after fitting are 95.59%, 85.58%, and 64.45%. Priced selection is the strongest or tied encoder through most of the range and pulls away from magnitude pruning as compression increases. Dense PCA remains substantially worse after fitting at the same 50% point (81.58%, 65.37%, and 30.80%), consistent with reconstruction quality alone not controlling the nonlinearity error.

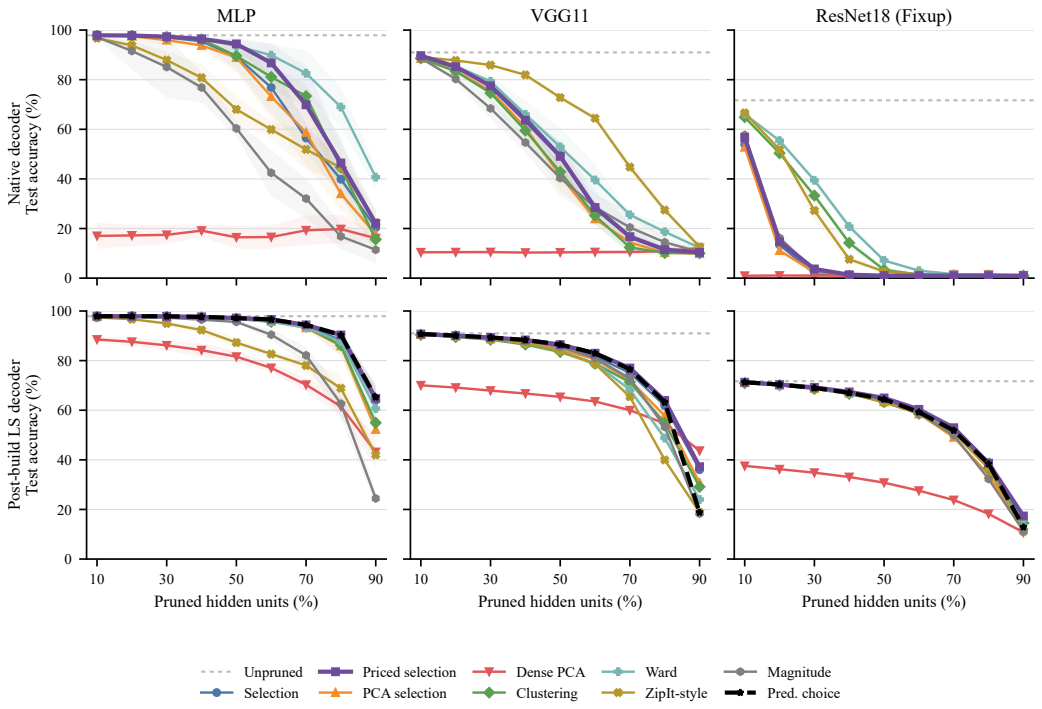


Figure 3: Single-model structured pruning across compression ratios (test accuracy %, mean over three source models; bands show sample standard deviations). The top row uses each encoder's native decoder, while the bottom row fits the same post-build least-squares decoder for every encoder. "ZipIt-style" denotes the single-model adaptation of its greedy correlation-merging rule, not the published two-model baseline.

The prediction-based choice usually uses priced selection at moderate compression and consequently tracks its accuracy. Its linearization is not reliable at the most severe settings: at 90% pruning on VGG11 it selects the ZipIt-style encoder and reaches 18.81%, whereas priced selection reaches 37.24% and dense PCA 43.67%. On ResNet18 it reaches 12.87% versus 17.36% for priced selection. We therefore treat prediction under extreme compression as a limitation.

C.3 Split-task merging under a matched calibration budget

The same-task comparison in Table 2 is not ZipIt's original setting. We therefore train three new pairs of specialists per architecture, with model A trained on the even classes and model B on the odd classes. We use BatchNorm MLPs on MNIST and BatchNorm VGG11 and ResNet18 on CIFAR-10. The MLPs use Adam at 10^{-3} for 25 epochs. VGG11 and ResNet18 use SGD with momentum 0.9, weight decay $5 \cdot 10^{-4}$, and cosine schedules starting at 0.05 and 0.1 for 100 and 160 epochs, respectively. All models use batch size 128, and we select each checkpoint by accuracy on a validation split of its five training classes, without using the test set.

Every method receives the same 2,000 unaugmented training examples drawn from the union of the two tasks. We evaluate on the full test set. For ZipIt, the 2,000 examples replace its full-training-set pass for covariance estimation and BatchNorm recalibration. We call the released matcher at commit `1b6ca40`. "Native tuned" selects (α, β) using the native decoder, while "decoder-aware tuned" selects them after fitting the decoder. Both choices use accuracy on the same calibration set. Only the selected candidate is evaluated on the test set. The fixed variant uses $(\alpha, \beta) = (10^{-4}, 0.075)$, and tuning searches

$$\alpha \in \{10^{-4}, .01, .1, .3, .5, 1\}, \qquad \beta \in \{0, .0375, .075, .125, .25, 1\}.$$

Table 9: Split-task test accuracy (%, mean ± sample sd over three independently trained pairs). "Known task" selects among the five classes of the input's task; "ten classes" selects among all classes. "+LS" fits only the final hidden/readout interface. The last two rows run both original models and are not compressed.

	MLP-BN		VGG11-BN		ResNet18-BN	
Method	known task	ten classes	known task	ten classes	known task	ten classes
Keep A, readout fit	94.18 ± 0.25	68.56 ± 3.58	75.68 ± 0.83	53.49 ± 1.06	84.62 ± 0.50	61.10 ± 1.74
Keep A, all-layer fit	94.10 ± 0.46	68.90 ± 3.67	75.67 ± 0.82	53.48 ± 1.01	84.52 ± 0.40	60.58 ± 1.60
ZipIt released	94.86 ± 0.42	73.69 ± 4.93	78.27 ± 0.19	54.58 ± 3.25	74.70 ± 1.95	46.89 ± 4.61
ZipIt released + LS	96.61 ± 0.42	82.78 ± 4.98	83.35 ± 0.59	66.35 ± 0.57	87.37 ± 0.77	72.30 ± 0.97
ZipIt tuned, native	97.04 ± 0.28	82.59 ± 2.64	85.00 ± 0.88	66.43 ± 0.99	81.15 ± 4.60	52.44 ± 11.05
ZipIt tuned, native + LS	97.34 ± 0.40	84.47 ± 3.49	85.78 ± 0.84	70.15 ± 1.12	88.50 ± 0.27	74.25 ± 0.69
ZipIt tuned after LS	97.31 ± 0.35	84.26 ± 3.19	85.54 ± 0.45	69.78 ± 0.71	89.16 ± 0.88	74.37 ± 1.28
Priced selection + LS	97.55 ± 0.43	75.97 ± 3.30	82.67 ± 0.16	62.70 ± 1.00	88.43 ± 2.38	68.68 ± 2.84
A's even scores + B's odd scores	98.86 ± 0.13	83.26 ± 4.65	92.75 ± 0.09	76.19 ± 0.58	95.50 ± 0.08	79.44 ± 1.53
Average A's and B's probabilities	98.86 ± 0.13	87.35 ± 2.44	92.75 ± 0.09	77.66 ± 0.19	95.50 ± 0.08	80.82 ± 1.08

The LS columns change only the final hidden/readout interface. After the merged BatchNorm statistics are fixed, a pooled post-build regression reconstructs both endpoint feature blocks from the realized merged feature on all 2,000 examples, and the resulting decoder is folded into the task-routed classifier. Intermediate interfaces retain the encoder's native decoder. The regression itself is label-free. Labels are used only by the two ZipIt tuning procedures.

Priced selection uses the same calibration examples for its statistics, BatchNorm recalibration, and final-interface fit without accuracy-based tuning. For the one-model controls, we keep A's hidden units and fit its readout to A's even-class and B's odd-class outputs. The second control also refits the hidden maps sequentially.

Both priced selection + LS and decoder-refitted ZipIt outperform the one-model all-layer fit on every pair under both metrics. Refitting released ZipIt increases mean known-task accuracy by 1.75, 5.08, and 12.67 points on MLP-BN, VGG11-BN, and ResNet18-BN, respectively. The corresponding ten-class gains are 9.09, 11.77, and 25.41 points. Encoder choice also matters: the tuned ZipIt rows exceed priced selection by 5.69–8.50 ten-class accuracy points, although priced selection is slightly higher on the MLP when the task is known.

C.4 Importance-priced selection

To compare selection rules, define the sample-dependent reconstruction residual $r_l(x) = (I - K_l' K_l) h_l^E(x)$. To first order, its effect on the output is

$$q_l(K_l' K_l) := \mathbb{E}_x \big\| J_l(x) r_l(x) \big\|^2, \qquad J_l(x) := \frac{\partial f^E(x)}{\partial h_l^E(x)}. \tag{17}$$

This quantity keeps each example's residual paired with that example's output sensitivity. It is label-free and uses the Jacobian of the uncompressed model E, but it must be evaluated separately for every candidate $K_l' K_l$. To obtain a reusable layer-wise price, we instead average the sensitivity first, defining the *propagation Gram*

$$G_l := \mathbb{E}_x \big[J_l(x)^\top J_l(x) \big], \qquad q_l^{\mathrm{avg}}(K_l' K_l) := \mathbb{E}_x \big[r_l(x)^\top G_l r_l(x) \big]. \tag{18}$$

Thus q_l^{avg} averages the output sensitivity separately from the reconstruction residual. It is exact when $J_l(x)^\top J_l(x)$ is constant, including at a fixed linear readout, and more generally when the sensitivity and residual are uncorrelated. The matrix G_l assigns a high cost to directions that strongly affect the output and a low cost to directions that the remaining network barely uses. On the vision models, we can estimate both objectives and test whether averaging the sensitivity loses useful information. For language models, we use the structured prices described in Section 6.3.

We test whether these prices improve a width-matched selection merge. At every hidden layer, the same greedy selection algorithm retains half of the stacked ensemble coordinates. Only its objective changes. We compare plain residual energy ($G = I$), the reusable propagation Gram G_l, and the sample-paired cost equation 17. We use 1,024 calibration examples for selection and 2,000 for the

Table 10: Importance-priced retained-channel selection with the post-build least-squares decoder. Each cell gives fused test accuracy (%) ± one sample standard deviation over three endpoint pairs, followed by the mean squared output deviation after "/". Bold independently marks the highest mean accuracy and the lowest mean deviation in each row.

Architecture	Unpriced ($G = I$)	Priced selection (G_l)	Paired $J_l(x)$
MLP-BN	98.10±0.08/12.45	98.15±0.05/**9.88**	**98.18**±0.12/10.20
MLP-noBN	97.95±0.07/12.28	**98.03**±0.06/**10.22**	97.96±0.05/10.33
VGG11-BN	**90.95**±0.10/8.45	90.83±0.12/**7.78**	90.91±0.12/8.64
VGG11-noBN	89.71±0.25/11.56	**90.07**±0.12/**9.67**	89.79±0.21/10.84
ResNet18-BN	72.91±0.15/61.16	**73.67**±0.02/**53.31**	73.38±0.34/56.58
ResNet18-Fixup	69.99±0.22/513.45	70.17±0.19/**478.33**	**70.21**±0.24/493.68

decoder fit and BatchNorm recalibration. Every encoder uses the post-build least-squares decoder. Table 10 reports means over three disjoint endpoint pairs.

Priced selection improves accuracy over the unpriced rule on five of six architectures and reduces the output deviation on all six. The sample-paired objective improves accuracy over priced selection on three architectures but does not reduce the output deviation on any of them. We therefore use the G_l-weighted rule as priced selection throughout the paper and include it as the seventh candidate for the prediction-based choice in Table 2.

C.5 Prediction diagnostics

BatchNorm recalibration changes the compressed model and therefore cannot be captured by a predictor that only uses the source models' running statistics. The predictor in Table 1 instead uses the recalibrated statistics of each candidate. For the clustering encoder on VGG11-BN, retaining the source models' running statistics gives a mean squared output deviation of 364.4 and a mean accuracy of 14.0%. Re-estimating the fused model's BatchNorm means and variances reduces the deviation to 27.6 and raises accuracy to 84.6%. On ResNet18-BN, the same recalibration reduces the mean squared output deviation from $85{,}393$ to 200.6 and raises accuracy from 1.5% to 43.4%.

We examine the prediction approximation with three additional tests. On a network with linear activations it is exact to floating-point precision for every tested encoder, as Proposition 9 requires. The linearized predictor can fail when an encoder densely mixes features across ReLU boundaries. For dense PCA encoders, it predicts only 7%–74% of the measured output deviation. Propagating the error through the actual ReLU instead of a fixed linearization reproduces the measured deviation almost exactly. Coordinate-selection encoders rarely change the ReLU activation pattern, so the linearized predictor already captures about 95% of their measured deviation.

C.6 Which error a method makes

The error decomposition splits the local error contributed at each layer into reconstruction and commutation error. For every encoder and endpoint pair, we report the commutation share

$$\frac{\sum_l \mathbb{E}\|\varepsilon_l^{\mathrm{com}}(x)\|^2}{\sum_l \mathbb{E}\|\varepsilon_l^{\mathrm{rec}}(x)\|^2 + \sum_l \mathbb{E}\|\varepsilon_l^{\mathrm{com}}(x)\|^2}.$$

The values below are the mean ± sample standard deviation of this share over the 18 endpoint pairs at the original target width.

Selection scores 0.000 ± 0.000, as it commutes with the elementwise nonlinearities (see Lemma 5). OT fusion scores 0.19 ± 0.10, permutation 0.28 ± 0.14, ZipIt 0.29 ± 0.13, and clustering 0.32 ± 0.16. CCA, the densest alignment, scores 0.56 ± 0.16 and reaches 0.92 on an MLP-without-BatchNorm pair. Dense invertible alignments therefore spend more than half of their local error budget on failing to commute with the activation.

C.7 MERGING MANY MODELS

Merging m models of width n into one model of the same width compresses an mn-dimensional ensemble representation to n dimensions. At layer l, let ν_i denote the eigenvalues of the uncentered second moment $\Sigma_l = \mathbb{E}[h_l^E(x)(h_l^E(x))^\top]$ of the concatenated ensemble activations. We use the unpriced ($G = I$) version of Lemma 4 and report the normalized reconstruction floor

$$100 \, \frac{\sum_{i>n} \nu_i}{\sum_i \nu_i} \, .$$

This is the percentage of the representation's total squared activation energy that even the best rank-n linear encoder–decoder cannot reconstruct at that layer. It is a local, method-independent quantity that disregards the commutation error. We estimate it from calibration activations before building a merged model.

In our experiments the normalized floor increases with m on every architecture, but each added model contributes fewer new directions than the previous one. The floor also varies across layers. When merging two VGG11 models, it is 12.0% after the 256-wide conv4 layer and 0.02% after the 512-wide conv6 layer.

The floor measures unavoidable local reconstruction error but does not determine downstream accuracy by itself. Table 11 shows that four MLPs can be merged without losing accuracy relative to the best single model. For VGG11 and Fixup ResNet18, the corresponding losses are about 1.8 and 3.1 points. The same-index average assumes that the models are already aligned and becomes substantially worse as more models are added.

Table 11: Multi-model merging and the normalized local reconstruction floor (accuracy %, mean over disjoint model subsets). The floor is measured after fc2 (128 channels) for the MLP, conv4 (256 channels) for VGG11, and block7 (512 channels) for ResNet18. All merges use the post-build refit. "Selection" retains n individual coordinates from the mn-dimensional ensemble representation; "same-index average" averages equally indexed channels across models without first aligning them.

Architecture	m	local floor %	selection	best single	ensemble	same-index avg.
MLP (no BN)	2	1.2	98.0	97.9	98.3	94.3
	3	2.0	97.8	97.9	98.4	90.3
	4	2.6	97.9	97.8	98.5	87.7
VGG11 (no BN)	2	12.0	89.7	90.9	91.6	82.0
	3	16.6	89.2	91.0	92.0	65.1
	4	19.5	89.2	91.0	92.1	43.6
ResNet18 (Fixup)	2	5.1	70.2	71.9	73.5	64.2
	3	7.3	69.5	71.9	74.0	60.1
	4	8.6	68.9	72.1	74.6	55.6

D ADDITIONAL LLAMA EXPERIMENTS

D.1 EVALUATION AND CALIBRATION PROTOCOL

We evaluate perplexity on the WikiText-2 test split (Merity et al., 2017) using non-overlapping windows of length 4096. The downstream score is the mean of PIQA, WinoGrande, HellaSwag, ARC-Easy, ARC-Challenge, and OpenBookQA. We use normalized accuracy where the harness provides it and raw accuracy for WinoGrande. All rows are zero-shot and use no recovery fine-tuning. Our models are evaluated with lm-evaluation-harness v0.4.12 (Sutawika et al., 2026) at batch size eight. The structurally modified FLAT-LLM and MoDeGPT implementations require batch size one; we verified batch-size equivalence on models that support both. Published rows are explicitly marked and are not presented as in-harness measurements.

Our default statistics use 192 calibration windows of length 2048, and the sequential feedforward fit uses 128 of them. "Wiki" uses WikiText-2 throughout. "Independent QA" replaces 48 of the 192

statistics windows by question–answer text drawn from the SciQ, QASC, and Social IQA training corpora, which do not belong to the six evaluation-task families. All independent-QA rows use an exact 25% mixture for both consumers: the first 128 windows contain 32 QA and 96 WikiText windows, and the remaining 64 contain 16 QA and 48 WikiText windows. The question–answer renderings contain the correct answer, although no supervised loss or parameter gradient is used. For Llama-2-13B only, the output-sensitivity statistics use 512-token subsequences to remain within GPU memory.

D.2 Separating reconstruction and query–key selection

We isolate the reconstruction and query–key changes in a dedicated ablation on Llama-2-7B. All four configurations use WikiText-only calibration, FLAT-LLM's released per-layer profile shape, the same Wanda feedforward selection and decoder, the original query–key normalization, and a matched total parameter fraction of approximately 0.8003. The no-query–key-pruning configurations spend the released parameter budget on the value–output and feedforward paths instead.

Table 12: Reconstruction $\times$ rotary-pair query–key selection on Llama-2-7B at matched parameter count (mean $\pm$ sample sd over three independent builds). "FLAT" uses head-wise activation PCA and its structural decoder; "ours" uses the output-metric basis and joint least-squares decoder.

Reconstruction	query–key	ppl $\downarrow$	avg. acc $\uparrow$
FLAT	dense	6.848 ± 0.003	57.42 ± 0.41
Ours	dense	6.399 ± 0.004	58.78 ± 0.38
FLAT	rotary pairs	6.573 ± 0.003	59.64 ± 0.12
Ours	rotary pairs	$\mathbf{6.293 \pm 0.002}$	$\mathbf{60.08 \pm 0.12}$

Changing only the reconstruction improves perplexity by 0.449 without query–key pruning and by 0.280 with rotary-pair selection. Adding rotary-pair selection improves perplexity by 0.275 with FLAT's reconstruction and by 0.106 with ours, with the corresponding accuracy gains being 2.22 and 1.30 points. The factors interact, so these differences are not additive, but each change improves both metrics with either setting of the other factor. The corresponding construction in Table 3 was evaluated in a separate set of three builds. It gives $60.19 \pm 0.09\%$ accuracy rather than $60.08 \pm 0.12\%$ here, while both report 6.293 ± 0.002 perplexity.

D.3 Calibration-only selection of the Llama-3 axis split

To select the Llama-3 attention multiplier without test leakage, we split the WikiText-2 training token stream once. The first 80% supplies the construction statistics and the final 20% supplies 64 held-out loss windows used only to rank the four candidates. This selects 1.30:

attention multiplier	1.00	1.15	1.30	1.45
held-out train ppl	9.879	9.748	**9.699**	9.703

Using this pruning ratio, three independently constructed models obtain 8.051 ± 0.030 test perplexity and $60.56 \pm 0.51\%$ six-task accuracy at parameter fraction 0.8004. Thus the attention-protecting split used in the main experiment is selected from calibration data alone. These constructions use only the first 80% of the training stream. Table 3 reports the same frozen split with the standard full WikiText calibration pool, yielding $8.070 \pm 0.021 / 60.98 \pm 0.32\%$.

Table 13 adds the in-harness baselines and the calibration-distribution ablation. The Wiki and independent-QA rows for 7B and 8B use the same driver, seeds, and construction. Independent-QA calibration improves downstream accuracy while slightly increasing perplexity. These rows therefore show a calibration-distribution trade-off rather than an additional method comparison.

D.4 Scope of the present allocation result

The 7B and 8B comparisons in Table 3 keep FLAT-LLM's released per-layer allocation. The 7B rows in Table 13 instead use a uniform per-layer allocation with value-head water-filling and are

Table 13: Complete structured-pruning results. Parameter fraction is relative to the dense checkpoint. N counts independently built compressed models; uncertainties are sample standard deviations. The Wiki and independent-QA rows for 7B and 8B are same-driver calibration controls. The Llama-2-13B FLAT-LLM row is published and cross-harness, whereas our Wiki row contains three in-harness builds; the main text therefore uses it as scale-up evidence rather than a controlled matched-harness ablation. The independent-QA rows are calibration- distribution ablations and should not be read as method improvements.

Model	method	calibration	N	params	ppl ↓	avg. acc ↑
Llama-2-7B	dense	–	1	1.0000	5.118	64.36
	SliceGPT (faithful, no recovery)	Wiki	1	0.8005	9.032	43.09
	FLAT-LLM (in harness)	Wiki	1	0.7998	6.415	59.44
	FLAT-LLM (published)	Wiki	1	~ 0.80	6.34	60.16
	MoDeGPT (in harness)	Wiki	1	0.8052	6.199	59.50
	ours (uniform layers)	Wiki	3	0.8004	5.895 ± 0.003	61.38 ± 0.55
	ours (uniform layers)	independent QA	3	0.8004	5.961 ± 0.008	62.60 ± 0.25
Llama-2-13B	dense	–	1	1.0000	4.575	67.17
	FLAT-LLM (published)	Wiki	1	~ 0.80	5.55	64.38
	ours	Wiki	3	0.8009	5.181 ± 0.002	64.70 ± 0.59
	ours	independent QA	3	0.8009	5.226 ± 0.008	65.49 ± 0.22
Llama-3-8B	dense	–	1	1.0000	5.755	68.37
	FLAT-LLM (parameter-matched)	Wiki	1	0.8009	8.454	60.32
	ours (FLAT profile, $t_a = 1.30$)	Wiki	3	0.8004	8.070 ± 0.021	60.98 ± 0.32
	ours (FLAT profile, $t_a = 1.30$)	independent QA	3	0.8004	8.162 ± 0.029	63.05 ± 0.34

therefore a different construction. For Llama-3, the attention multiplier is selected by held-out loss on WikiText-2 training data (Appendix D.3), while Llama-2 uses the proportional split.

A calibration-only native allocator performed poorly: its single Llama-2-7B run at parameter fraction 0.8003 reached 26.115 perplexity and 52.27% six-task accuracy, compared with 5.895 ± 0.003 and $61.38 \pm 0.55\%$ for the reported Wiki-only construction. Thus the current local calibration prices do not yet provide a competitive native allocation rule. At 13B, our method has three in-harness builds at parameter fraction 0.8009, but the released FLAT-LLM QK and no-QK paths both fail for this rotary-attention implementation. We therefore report its published numbers.

E Methods as Generalized Pruning

We express training-free merging and pruning methods from the literature as kernel pairs (K_l, K'_l), verifying in each case that the induced weights $W_l^S = K_l W_l^E K'_{l-1}$ reproduce the method as published. Recall the setting of Section 2: the large model E has widths m_l, the compressed model S has widths $r_l \le m_l$, and in the merging case features are ordered so that $h_l^E(x) = \left[\begin{smallmatrix} h_l^A(x) \\ h_l^B(x) \end{smallmatrix}\right]$, $m_l = 2n_l$. We write $\bar{\lambda} := 1 - \lambda$ and partition the second moment into blocks $\Sigma_l = \left[\begin{smallmatrix} \Sigma_l^{AA} & \Sigma_l^{AB} \\ \Sigma_l^{BA} & \Sigma_l^{BB} \end{smallmatrix}\right]$, so that Σ_l^{AB} is the cross-covariance on which matching methods operate. The decoder K'_l is never materialized at inference: it is folded into the next layer's weights, so S has the standard architecture at the target widths. Table 14 summarizes the catalogue.

Permutation alignment. (Entezari et al., 2022; Ainsworth et al., 2023) The permutation kernels are given in Section 2. The permutation can be found by weight matching, straight-through estimation (Ainsworth et al., 2023), activation matching on correlations (Li et al., 2016), including Hungarian assignment (Tatro et al., 2020), or implicit differentiation through Sinkhorn (Guerrero Peña et al., 2023). For positively homogeneous activations, the kernel can also include a positive diagonal rescaling: $K_l = \begin{bmatrix} \lambda I & \bar{\lambda} P_l \Lambda_l \end{bmatrix}$, $K'_l = \left[\begin{smallmatrix} I \\ \Lambda_l^{-1} P_l^\top \end{smallmatrix}\right]$ with $\Lambda_l \succ 0$ diagonal.

Optimal transport fusion. (Singh & Jaggi, 2020) Replace P_l by $n_l T_l$, where T_l is a coupling with uniform marginals, so that $n_l T_l$ is doubly stochastic and reduces to a permutation exactly for hard couplings: $K_l = \begin{bmatrix} \lambda I & \bar{\lambda}\, n_l T_l \end{bmatrix}$, $K'_l = \left[\begin{smallmatrix} I \\ n_l T_l^\top \end{smallmatrix}\right]$. For hard couplings this is the permutation pair above and for soft couplings $K_l K'_l = \lambda I + \bar{\lambda}\, n_l^2\, T_l T_l^\top \neq I$: the transposed plan is not a right

Table 14: Methods as kernel choices in the generalized pruning framework.

Method	K_l	K'_l
Ensemble	I	I
Weight averaging	$\begin{bmatrix}\lambda I & \bar{\lambda} I\end{bmatrix}$	$\begin{bmatrix} I \\ I \end{bmatrix}$
Permutation (Ainsworth et al., 2023)	$\begin{bmatrix}\lambda I & \bar{\lambda} P_l\end{bmatrix}$	$\begin{bmatrix} I \\ P_l^\top \end{bmatrix}$
OT fusion (Singh & Jaggi, 2020)	$\begin{bmatrix}\lambda I & \bar{\lambda} n_l T_l\end{bmatrix}$	$\begin{bmatrix} I \\ n_l T_l^\top \end{bmatrix}$
ZipIt (Stoica et al., 2024)	$D_l^{-1} H_l^\top$ (cluster average)	H_l (membership copy)
CCA-Merge (Horoi et al., 2024)	$\begin{bmatrix}\lambda T_l^A & \bar{\lambda} T_l^B\end{bmatrix}$	$\begin{bmatrix} (T_l^A)^{-1} \\ (T_l^B)^{-1} \end{bmatrix}$
Selection pruning (Li et al., 2017)	R_l	$R_l^\top$
OBC-style refit (Frantar & Alistarh, 2022)	R_l	$\Sigma_l R_l^\top (R_l \Sigma_l R_l^\top)^{-1}$
PLeaS (Nasery et al., 2025)	$K_l^{\text{perm/partial}}$	endpoint-activation LS (Adam)
SliceGPT (Ashkboos et al., 2024)	$Q_l^\top$	Q_l

inverse of the encoder. Encoding after decoding therefore changes the compressed features and blends information across units.

ZipIt. (Stoica et al., 2024) Let $H_l \in \{0,1\}^{m_l \times r_l}$ be the membership matrix returned by ZipIt's greedy matcher and $D_l = \operatorname{diag}(H_l^\top \mathbf{1})$ the cluster-size matrix. Its released kernels are $K_l = D_l^{-1} H_l^\top$ and $K'_l = H_l$: merge by averaging each cluster and unmerge by copying its value to every member. The repeated-merge penalty α and within-model budget β permit clusters larger than two and clusters containing several features from the same model. Consequently, K'_l is not generally $2K_l^\top$, although $K_l K'_l = I$. A disjoint perfect pairing is only the special case in which every cluster has size two. Partial zipping corresponds to $K_l = K'_l = I$ beyond the zip depth.

CCA-Merge and invertible alignments. (Horoi et al., 2024) $K_l = \begin{bmatrix}\lambda T_l^A & \bar{\lambda} T_l^B\end{bmatrix}$, $K'_l = \begin{bmatrix} (T_l^A)^{-1} \\ (T_l^B)^{-1} \end{bmatrix}$, $r_l = n_l$, with T_l^A, T_l^B invertible transforms into a common basis computed from the canonical directions of the whitened cross-covariance $(\Sigma_l^{AA})^{-1/2} \Sigma_l^{AB} (\Sigma_l^{BB})^{-1/2}$. These kernels satisfy $K_l K'_l = I$.

Reconstruction-aware pruning and refitting. (He et al., 2017; Luo et al., 2017) Keep the selection encoder $K_l = R_l$ but replace the transpose by the decoder $\Sigma_l R_l^\top (R_l \Sigma_l R_l^\top)^{-1}$. Folded into the next layer, $W_{l+1}^S = K_{l+1} W_{l+1}^E K'_l$ then reconstructs the full features from the retained ones before applying the next layer. This includes the reconstruction step of optimal brain compression (Frantar & Alistarh, 2022). The same decoder appears as Nyström reconstruction in MoDeGPT (Lin et al., 2025) and FLAT-LLM (Tian et al., 2026, App. C), which use ridge leverage scores to choose the retained feedforward neurons. PLeaS (Nasery et al., 2025) instead combines a permutation or partial-matching encoder with an activation reconstruction objective that its released implementation optimizes with Adam.

Spectral truncation. (Ashkboos et al., 2024) $K_l = Q_l^\top$, $K'_l = Q_l$, with $Q_l \in \mathbb{R}^{m_l \times r_l}$ the top-r_l eigenvectors of Σ_l. SliceGPT applies this pair along the residual stream of a transformer, using the invariance of RMSNorm under orthogonal transforms (Ashkboos et al., 2024) to absorb Q_l into adjacent weights; the architectural details are given in Appendix G.

Scope remarks. (i) *Biases.* With biases, tracking requires $b_l^S = K_l b_l^E$. Equivalently, append a constant feature and absorb the bias column into W_l^E, in which case the conventions above apply verbatim. (ii) *More than two models.* For k models, W_l^E is block-diagonal with k blocks, the readout concatenates $\lambda_i W_L^{M_i}$ with $\sum_i \lambda_i = 1$, K'_0 stacks k identities, and all merging kernels above extend with k column blocks. (iii) *Rank compression.* Methods that factorize weight matrices into low-rank products compress rank rather than width. They can be expressed as kernel pairs acting on a virtual linear layer, but they are outside the scope of our experiments.

F PROPAGATION DETAILS

This appendix gives the details behind Section 5: exact propagation from cached activations of E, its linearized approximation, ReLU flip analysis, and extensions to BatchNorm, refitted decoders, and residual architectures. The exact recursion evaluates a candidate's output deviation without constructing the corresponding compressed model.

Proposition 8. Let $\hat{\varepsilon}_1(x) := 0$ and, for $l \in \{1, \dots, L-1\}$,

$$\hat{\varepsilon}_{l+1}(x) := K_{l+1} W_{l+1}^E \Big(h_l^E(x) - K_l' \, \sigma_l\big(K_l z_l^E(x) - \hat{\varepsilon}_l(x)\big)\Big). \tag{19}$$

Then $\hat{\varepsilon}_l(x) = \varepsilon_l(x)$ for every l and in particular $\hat{\varepsilon}_L(x) = \varepsilon_L(x)$.

Proof. Induction: $\hat{\varepsilon}_1(x) = \varepsilon_1(x) = 0$, and if $\hat{\varepsilon}_l(x) = \varepsilon_l(x)$ then $\sigma_l(K_l z_l^E(x) - \hat{\varepsilon}_l(x)) = \sigma_l(z_l^S(x)) = h_l^S(x)$, so the right-hand side of equation 19 equals $\varepsilon_{l+1}(x)$ by Lemma 6. □

All quantities in equation 19 are obtained from one cached forward pass of E and the candidate kernels. Expanding through $\pm K_l' K_l h_l^E(x)$ and $\pm K_l' \sigma_l(K_l z_l^E(x))$ recovers the three terms of Theorem 2. The identity is exact for the model assembled by equation 2. Candidate-specific changes made afterward, such as BatchNorm recalibration or post-build decoder fitting, require the corresponding updated statistics. Expected errors are accurate only insofar as the calibration set represents the evaluation distribution.

BatchNorm-aware transport. For BatchNorm models, we re-estimate each candidate's running statistics and fold the resulting affine maps into the recursion. This requires constructing the candidate far enough to collect its statistics. The affine form is given in Appendix G, and the synchronization procedure is tested in Appendix C.1.

Linearized propagation. Replacing the transport by its Jacobian along E's trajectory, with $\zeta_l := K_l z_l^E(x)$ and $D_l = \mathrm{diag}(\sigma_l'(\zeta_l))$ as in Section 5, gives the recursion

$$\hat{\varepsilon}_{l+1}(x) = \varepsilon_{l+1}^{\mathrm{rec}}(x) + \varepsilon_{l+1}^{\mathrm{com}}(x) + W_{l+1}^S \, D_l \, \hat{\varepsilon}_l(x), \tag{20}$$

whose unrolling from $\hat{\varepsilon}_1(x) = 0$ is the superposition equation 15 of the main text.

Proposition 9 (Exactness for linear activations)**.** If $\sigma_l(t) = a_l t$, then $\varepsilon_{l+1}^{\mathrm{com}}(x) = 0$ for every kernel pair, $D_l = a_l I$, and the transport in equation 20 coincides with the exact transport of Theorem 2. In particular, if every σ_l is linear, the linearized propagator reproduces the output deviation exactly.

Proof. $K\sigma(u) = aKu = \sigma(Ku)$ for any K, and $\sigma_l(\zeta_l) - \sigma_l(\zeta_l - \varepsilon_l(x)) = a_l \varepsilon_l(x) = D_l \varepsilon_l(x)$. □

For nonlinear σ_l the defect is confined to the transport: the remainder $R_l(\varepsilon) := \sigma_l(\zeta_l) - \sigma_l(\zeta_l - \varepsilon) - D_l\varepsilon$ vanishes coordinatewise for ReLU unless the perturbation flips the sign of the pre-activation, and satisfies $|R_l(\varepsilon)_j| \le |\varepsilon_j| \, \mathbf{1}\{|(\zeta_l)_j| \le |\varepsilon_j|\}$. The propagator is therefore reliable while the tracked error is small relative to the pre-activation margins, and the *flip mass* $\mathbb{E} \sum_j \varepsilon_j^2 \, \mathbf{1}\{|(\zeta_l)_j| \le |\varepsilon_j|\}$, which can be estimated during the same recursion, certifies this per method. It measures the tracked error against the pre-activation margins rather than the commutation error alone. Selection kernels can still carry flip mass through propagated reconstruction error even though their commutation error vanishes. When the flip mass is large, Proposition 8 provides an exact nonlinear fallback without constructing and evaluating the candidate network.

Refitted decoders. To evaluate a refit, substitute the fitted decoder $K_l'^{\star}$ (and its intercept) for K_l' in the creation and transport terms of equation 19 or equation 20. A post-build decoder depends on the activations of the partially constructed model and can compensate for upstream error.

Residual streams. On a residual architecture the recursion follows the computational graph. An identity skip carries the current tracking error unchanged, while the residual branch contributes its local and propagated errors. A sequential layer walk that omits the skip-carried term is therefore incomplete. The construction conventions are given in Appendix G.

G Architecture Extensions

For convolutions and transformer blocks, the kernels act on channel or hidden coordinates rather than on flattened tensors. This section states the construction rules needed for the architectures in our experiments.

G.1 Convolution and pooling

For a convolution, kernels act on the input and output channel dimensions. If $\mathcal{W}_l^E(:,:,q)$ is the input–output channel matrix at spatial kernel offset q, then the encoder–decoder sandwich is applied independently at every offset:

$$\mathcal{W}_l^S(:,:,q) = K_l\, \mathcal{W}_l^E(:,:,q)\, K'_{l-1}. \tag{21}$$

Convolutional biases follow the constant-feature convention of Appendix E.

Max pooling acts independently in each channel. It commutes with a nonnegative rescaled selection or permutation row because $\max_q cu_q = c \max_q u_q$ for $c \geq 0$, but not with a row that blends channels. In the propagator, the linearized error is routed through the argmax selected on the reference activation. A changed argmax in the pooling layer is analogous to a ReLU sign flip and is handled by exact nonlinear transport when needed.

G.2 Normalization layers

At inference, BatchNorm is the channel-wise affine map

$$\mathrm{BN}(u) = s \odot u + t, \qquad s = \frac{\gamma}{\sqrt{v+\epsilon}}, \qquad t = \beta - s \odot \mu, \tag{22}$$

where (μ, v) are its running statistics. Once these statistics are fixed, the map can be folded exactly into adjacent affine operations and creates no nonlinear commutation term. Merging changes the input distribution, however, so the source models' stored statistics need not describe the fused model. Appendix C.1 explains and tests the recalibration used for post-build decoder fitting.

LayerNorm and RMSNorm couple feature coordinates through their sample-wise statistics and therefore cannot in general be treated as coordinatewise nonlinearities. For gain-free RMSNorm, a square orthogonal change of basis Q commutes without any error, $\mathrm{RMSNorm}(Qu) = Q\,\mathrm{RMSNorm}(u)$. These invariances apply only to square changes of basis, not directly to a width reduction. A learned RMSNorm gain scales each coordinate separately and therefore does not commute with an arbitrary rotation. It must instead be absorbed into an adjacent linear layer or transformed together with the basis. LayerNorm imposes an additional constraint because it subtracts the mean across features: an orthogonal transformation commutes with this centering operation only if it preserves the constant-vector direction. Finally, selecting a rectangular subset of coordinates changes both the features and the dimension over which the normalization statistic is computed. Thus these arguments justify certain orthogonal reparameterizations, such as the one used by SliceGPT, but not arbitrary pruning of the residual stream.

G.3 Residual streams

All branches that meet at a residual addition must produce features in the same compressed coordinate system. Appendix F gives the corresponding error recursion.

G.4 Attention and gated feedforward blocks

The value–output path of an attention module is linear in the value features: V is aggregated across tokens and then read by the output projection. Dense mixing is therefore admissible when accompanied by a matching decoder or an output-projection refit, and a fixed total value rank can be distributed across heads.

The query–key path is different. Its compressed features determine the bilinear scores passed through softmax, so a generic dense kernel has no coordinatewise commutation guarantee. Our

Llama construction therefore selects the same coordinates for queries and keys and restricts the keep set to complete RoPE pairs. Each pair is a two-dimensional invariant subspace of every rotary transformation; selecting only one coordinate would make the compressed rotary map position dependent.

A SwiGLU hidden unit is likewise a coupled object: one gate row, the corresponding up-projection row, and the matching down-projection column. Coordinate pruning must use one keep set for all three. Dense mixing does not commute with the SiLU gate or elementwise product, whereas selection does. The down projection can then again be refitted as the decoder.