SYMMETRY AND OPTIMIZATION IN OVERPARAMETERIZED NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Overparameterization is central to the success of deep learning, yet the mechanisms by which it improves optimization remain incompletely understood. In hopes of better understanding how the loss landscape transforms as we add neurons, we study neural networks of different width that represent the same function. Building on prior work, we characterize function-preserving transformations across network width in terms of families of matrices. These width symmetries allow for the study of a hierarchy of solutions across neural network width. We then use this construction to derive consequences for loss landscape geometry and optimization. First, we prove that these symmetries act as a form of diagonal preconditioning on the Hessian, enabling the existence of better-conditioned minima within each class of functionally identical solutions. Second, we find that overparameterization increases the probability mass of global minima near typical initializations, making these favorable solutions more reachable. These results offer a potential link between loss landscape geometry and simplicity bias. Empirically, we observe wider networks have lower top eigenvalues, smaller condition numbers and faster convergence, supporting our analysis. Together, these results describe width growth as a geometric transformation of the loss landscape.

1 INTRODUCTION

Overparameterized neural networks consistently outperform their smaller counterparts in optimization speed and final performance, even when the target function could be represented by much smaller architectures (Arora et al., 2018; Yang et al., 2022). A major line of theory explains this in the infinite limit via linearization (Jacot et al., 2018; Allen-Zhu et al., 2019; Du et al., 2019; Zou et al., 2019).

At finite widths, the loss landscape is also shaped by *symmetry*: neural networks possess large groups of function-preserving reparameterizations (permutations, rescalings, and more structured invariances) that induce equivalence classes of weights and manifolds of functionally identical minima (Simsek et al., 2021; Kunin et al., 2021; Grigsby et al., 2023; Martinelli et al., 2024).

At the intersection of these perspectives, we study the impact of adding neurons at finite width. In particular, we ask: *"How do the solutions of small networks relate to the solutions of large networks?"* Alternatively, if we have some information about the loss landscape of narrow networks, what can we say about the landscape of wider networks?

Building on Fukumizu & Amari (2000); Simsek et al. (2021), we characterize weight-space symmetries across width and show their impact on optimization through two mechanisms:

Diagonal Preconditioning. The eigenvalues of the Hessian H after a symmetry transformation are the same as those for $K^{1/2}HK^{1/2}$, where K is a diagonal matrix that depends on the transformation. The right choice of K can improve the Hessian conditioning.

Volume Growth. Overparameterization increases the total volume of global minima near typical initializations. Specifically, we compute how the probability that a random initialization lands in the ϵ-sublevel set of a global minimum changes when a neuron is added to a minimal-width network.

Together, these mechanisms provide a constructive view of overparameterization through a hierarchy of functionally equivalent solutions across widths, allowing us to track how their geometry changes as neurons are added. Unlike linearization-based analyses, the framework applies at finite width and can therefore provide insights even in the underparameterized regime.

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.

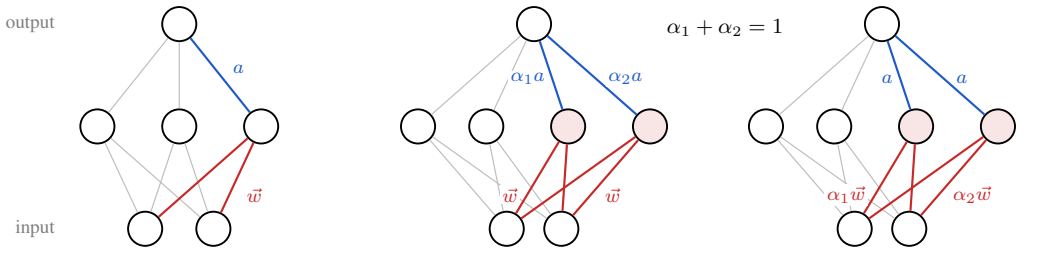


Figure 1: Splitting a hidden neuron. Input is at the bottom, output at the top; the neuron with incoming weights $\vec{w}$ and outgoing weight a is split into two copies (shaded). Post-activation splitting copies the incoming weights and divides the outgoing one, and works for any activation. Pre-activation splitting divides the incoming weights and copies the outgoing one, and works for activations that are equivariant to positive scaling, such as ReLU. With more outputs, each outgoing weight is split the same way.

2 Preliminaries

Parameter Space and Loss. Fix a depth-L feedforward architecture with layer widths $n_0 = d_{\text{in}}, n = n_1, \ldots, n_{L-1}, n_L = d_{\text{out}}$. Let Θ denote the corresponding parameter space (weights and biases). For $\theta \in \Theta$, write the realized function as $f_\theta : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$. Given a ground-truth function f^* and an input distribution $\mathcal{D}$, we consider the population risk $\mathcal{L}(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\big[\ell\big(f_\theta(x), f^*(x)\big)\big]$. Our focus is on *geometric* properties of $\mathcal{L}$ in parameter space near minima: equivalence classes of functionally identical parameters, their induced flat directions, and the curvature in directions transverse to these classes.

Functional Equivalence and Symmetry Orbits. We distinguish *functional equivalence* from *equivalence under a chosen symmetry family*. Define $\theta \sim_f \theta'$ if $f_\theta = f_{\theta'}$ (equality as functions), and define $\theta \sim_{\mathcal{T}} \theta'$ if there exists $g \in \mathcal{T}$ with $g(\theta) = \theta'$, where

$$\mathcal{T} = \{g : \Theta \to \Theta \mid f_\theta(x) = f_{g(\theta)}(x)\ \forall x, \theta\}. \tag{1}$$

Write the corresponding equivalence classes as

$$[\theta]_f = \{\theta' : \theta' \sim_f \theta\}, \qquad [\theta]_{\mathcal{T}} = \{\theta' : \theta' \sim_{\mathcal{T}} \theta\}.$$

For $\sim_{\mathcal{T}}$ to be an equivalence relation, we need $\mathrm{Id} \in \mathcal{T}$ and $g \in \mathcal{T} \iff g^{-1} \in \mathcal{T}$. Transitivity follows from composition. When $\mathcal{T}$ contains all function-preserving transformations, $[\theta]_{\mathcal{T}} = [\theta]_f$, otherwise $[\theta]_{\mathcal{T}} \subset [\theta]_f$. In our case, for special activations studied by Simsek et al. (2021), the operations in Section 3 generate all of $[\theta]_f$ at a global minimum, so the two classes coincide. In general, the full set of symmetries for arbitrary activations is not known.

Realizability. We operate in a realizable setting, assuming the target function f^* can be exactly represented by a neural network of some minimal (functional) width n_f. When an overparameterized network of width $n \geq n_f$ is trained to match f^*, global minima exist by construction. The central question is how this excess capacity reshapes the symmetry orbit structure and, through it, the curvature and reachability properties that govern optimization.

To make this setup mathematically explicit, we often adopt the terminology of a *teacher-student* framework. A *teacher* parameter vector $\theta^* \in \Theta_{n_f}$ defines the minimal target function $f^* = f_{\theta^*}$, while a *student* network of width n is trained to match it. However, this terminology is not restrictive: universal approximation theorems guarantee that any continuous function can be approximated to any desired accuracy by a network of some finite width, and we take the function at that width as the teacher.

3 Characterizing Overparameterization Symmetries

We begin by noting the definition of overparameterization used in this work, following Simsek et al. (2021). We say a neural network is overparameterized if it has greater width than required

to represent a given function. This is a natural notion when studying optimization toward a fixed function rather than over a dataset. It is related to but distinct from the notion broadly relating the number of parameters to data samples.

Prior work gives a description of the overparameterized solution set, primarily for single-hidden-layer networks. Here, we unify these results with a useful matrix viewpoint and explain what changes with depth and choice of activation.

3.1 Fixed-width Symmetries in Nonlinear Networks

We first remark that a number of well-studied fixed-width parameter symmetries can be seen as matrices. For a network with activation σ to have a symmetry via matrix M, we require

$$M^{-1}\sigma(Mx) = \sigma(x)\,\forall x. \tag{2}$$

For example, permuting hidden units in a width-n layer is obtained by taking M to be a permutation matrix. Similarly, for ReLU networks, neuron-wise rescaling is obtained by taking $M = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ with $\lambda_i > 0$. For permutations, Equation (2) holds since σ is applied pointwise, and for ReLU, the activation is equivariant to positive rescaling.

3.2 Overparameterization in Deep Linear Networks

Deep linear networks are parameterized as $f(x) = W_L \cdots W_2 W_1 x$. It is well documented that deep linear networks have a matrix factorization symmetry that impacts optimization (Saxe et al., 2014; Ghosh et al., 2025). In particular, if we have fixed width n, between any pair $W_{k+1}W_k$ we can insert an invertible matrix $M \in \mathrm{GL}_n(\mathbb{R})$ as $M^{-1}M$. This is the same symmetry present between W_K and W_Q in self-attention.

To generalize this across widths, we can think of M as an expansion followed by a contraction operation. Here, we have $M \in \mathbb{R}^{r\times n}$ for $r > n$ and a corresponding pseudoinverse $M^+ \in \mathbb{R}^{n\times r}$ such that $M^+M = I_n$. Due to the use of the pseudoinverse, rather than the inverse, we have additional degrees of freedom in the choice of M^+. The transformation $W_k \to MW_k$ and $W_{k+1} \to W_{k+1}M^+$ preserves the function while changing the parameterization.

3.3 General Activations: Post-Activation Splitting

In the presence of an arbitrary activation σ between layers, the matrices M become much more restricted since one cannot arbitrarily mix hidden units while preserving $f_\theta(x)$ for all x. If we make *no assumptions* about σ, two operations preserve the function for every choice of activation:

1. *Copy* the input and split the outgoing weights: duplicating a neuron with inputs $\vec{w}$ and outputs $\vec{a}$ into neurons $(\vec{w}, \vec{w})$ with outputs $(\vec{a}', \vec{a}'')$ satisfying $a'_i + a''_i = a_i$.
2. Add *0-type* neurons with arbitrary inputs and output weights summing to $\vec{0}$.

For a special class of activations, Simsek et al. (2021) show these are the complete set of function-preserving transformations.

We characterize splitting through a class of *duplication matrices*. Here, we allow expansion by a matrix $M \in \mathbb{R}^{r\times n}$ for $r > n$. M is a duplication matrix and repeats rows $\vec{e}_i$ of I_n with multiplicity m_i, $M = \begin{bmatrix}\vec{e}_1 \otimes \mathbf{1}_{m_1}^\top & \cdots & \vec{e}_n \otimes \mathbf{1}_{m_n}^\top\end{bmatrix}^\top$, where $\mathbf{1}_{m_i}$ is an m_i-dimensional vector of ones and $\otimes$ denotes the Kronecker product. Thus, each basis vector $\vec{e}_i$ of I_n is duplicated m_i times, yielding a total expansion $r = \sum_i m_i$.

Then, M^+ can be *any* pseudoinverse such that $M^+M = I_n$. The freedom in choosing M^+ corresponds to redistributing outgoing weights across the expanded layer, i.e., picking $(\vec{a}', \vec{a}'')$ so $a'_i + a''_i = a_i$ for all i. Our symmetry corresponds to transforming

$$W_l \to MW_l \text{ and } W_{l+1} \to W_{l+1}M^+.$$

We extend this construction with per-weight coefficient matrices C_i, which give a strictly richer symmetry than the per-neuron formulation in Simsek et al. (2021).

Example: Duplication and Pseudoinverse

For $M = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$ (duplicating the first neuron), we can have several pseudoinverses M^+:

$$M^+ = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 0.9 & 0.1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \text{etc.}$$

The freedom in M^+ corresponds to how we split the output weights.

3.4 Pre-activation Splitting

Many practical activations are not so pathological. For instance, ReLU is linear in half the domain, which makes it equivariant to positive rescaling: $\sigma(\lambda x) = \lambda\sigma(x)$ for $\lambda > 0$. This gives a second way to split a neuron, which we call *pre-activation splitting*.

To split the activations evenly, we multiply the pre-activation weights by positive coefficients $[d_1, d_2, \ldots, d_m]$ such that $\sum_i d_i = 1$. If the original pre-activation is positive, the ReLU remains linear, preserving the output. If negative, the activation stays negative after splitting.

This can be written as left-multiplying by the expansion matrix $M' = \begin{bmatrix} \vec{e}_1 \otimes D_{m_1}^\top & \cdots & \vec{e}_n \otimes D_{m_n}^\top \end{bmatrix}^\top$, where $D_{m_i} = [d_1, \ldots, d_{m_i}]^\top$ is a column vector with $d_j > 0$ and $\sum_j d_j = 1$. The corresponding contraction transformation on W_{l+1} involves copying the columns, which is represented by right-multiplying by the pseudoinverse $M'^+ = \begin{bmatrix} \vec{e}_1 \otimes \mathbf{1}_{m_1}^\top & \cdots & \vec{e}_n \otimes \mathbf{1}_{m_n}^\top \end{bmatrix}$. We note that this symmetry can be obtained for ReLU by applying a post-activation split and rescaling, though it additionally applies to a broader class of activations discussed in Appendix E.

A natural question is: what algebraic structure do these transformations correspond to? In Appendix F, we discuss why they make up a *groupoid* (Figure 7), an object similar to a group but with operations that need not be defined on the entire domain.

4 Landscape Geometry and Optimization

Remark 4.1. We remark that though we have defined particular symmetry matrices M in the above section, the results that follow are *fully agnostic* to the symmetry operation. In particular, if we take any M describing a symmetry, for example, permutation, rescaling, matrix factorization or any other operation from Appendix C or D, the gradients will transform according to $B^\top \nabla L(\theta)$, the Hessian as $B^\top H B$, the nonzero eigenvalues as those of HK and the pseudo-determinant as $\mathrm{pdet}\,(H)\det(K)$, with the only difference that K no longer need be diagonal.

4.1 Gradients Under Splitting

Theorem 4.2 (Gradient Transformation)**.** *Let θ' be the image of $\theta = \mathrm{vec}(W_1, \ldots, W_L)$ under layer-wise expansions (M_l, M_l^+). Then $\nabla L(\theta') = B^\top \nabla L(\theta)$ for the constant matrix*

$$B^\top = \mathrm{diag}\left(I \otimes (M_1^+)^\top,\ \ldots,\ M_{l-1} \otimes (M_l^+)^\top,\ \ldots,\ M_{L-1} \otimes I\right). \tag{3}$$

Concretely, for a split with coefficients $\alpha_1, \ldots, \alpha_m$ summing to one: each copy of a duplicated weight receives α_i times the original gradient, and every other gradient is unchanged.

4.2 Hessians Under Splitting

Let $H \in \mathbb{R}^{a\times a}$ be the Hessian of the original parameterization at a zero-loss minimum and let $H' \in \mathbb{R}^{b\times b}$ be the expanded Hessian obtained after applying an expansion from $a \to b$ weights. Differentiating the gradient at the minimum, we can express the expanded Hessian as $H' = B^\top H B$, where $B^\top \in \mathbb{R}^{b\times a}$ is the matrix that maps the gradient of the original network to the gradient of the expanded one. Even when not exactly at the minimum, this expansion still gives a good approximation when the additional Gauss-Newton residual is small (Tables 2 and 3).

Example: Hessian Transformation

For a 1-hidden-layer network with 1 hidden neuron, 1 input weight w and 1 output weight a, duplicating a and splitting w with coefficients $(\alpha, 1-\alpha)$ gives (writing $\bar{\alpha} := 1-\alpha$)

$$H = \begin{bmatrix} h_{ww} & h_{wa} \\ h_{wa} & h_{aa} \end{bmatrix} \;\to\; H' = \begin{bmatrix} h_{ww} & h_{ww} & \alpha h_{wa} & \bar{\alpha} h_{wa} \\ h_{ww} & h_{ww} & \alpha h_{wa} & \bar{\alpha} h_{wa} \\ \alpha h_{wa} & \alpha h_{wa} & \alpha^2 h_{aa} & \alpha\bar{\alpha} h_{aa} \\ \bar{\alpha} h_{wa} & \bar{\alpha} h_{wa} & \alpha\bar{\alpha} h_{aa} & \bar{\alpha}^2 h_{aa} \end{bmatrix}.$$

The two duplicated rows/columns are everywhere scaled by α and $\bar{\alpha}$ respectively, tracing back to the gradient scaling in Theorem 4.2.

4.3 Overparameterization as Diagonal Preconditioning

Theorem 4.3 (Diagonal Preconditioning). *Let $H \in \mathbb{R}^{a\times a}$ be the Hessian of the loss at a zero-loss minimum and $H' \in \mathbb{R}^{b\times b}$ the Hessian after a splitting. Then, H' has at least $b-a$ zero eigenvalues and its remaining eigenvalues are those of HK, equivalently of $K^{1/2}HK^{1/2}$, for $K = BB^\top \in \mathbb{R}^{a\times a}$. K is diagonal with entries m for each split weight, $\sum_{i=1}^{m}\alpha_i^2$ for each weight duplicated with proportions $\alpha_1, \ldots, \alpha_m$, and 1 for every other weight.*

The key point is that, in addition to introducing new flat directions, widening a network also systematically reshapes the curvature in the original parameter directions. In particular, splitting rescales different coordinates of the Hessian by different amounts, so moving to a functionally equivalent wider network is spectrally equivalent to applying a diagonal preconditioner to the original Hessian. The freedom in choosing the split coefficients therefore gives a family of functionally identical minima with different conditioning.

4.4 Local Volume and Pseudo-Determinant

The ϵ-sublevel set of a minimum θ^* is locally an ellipsoid with volume $\propto (\det H)^{-1/2}$. Since splitting introduces flat symmetry directions, H' is singular, so we measure volume *modulo symmetry directions* via the pseudo-determinant $\mathrm{pdet}(H') = \prod_{i:\lambda_i>0}\lambda_i$. By Theorem 4.3, the nonzero eigenvalues of H' are exactly those of HK, so when H is full rank,

$$\mathrm{pdet}(H') = \det(HK) = \mathrm{pdet}(H)\det(K).$$

Theorem 4.4 (Pseudo-determinant Under Splitting). *Let H be full rank. Split one neuron m ways with coefficients $\alpha_1, \ldots, \alpha_m$ summing to 1, and let S and D be the sets of split and duplicated weights. Then*

$$\mathrm{pdet}(H') = m^{|S|}\Big(\sum_i \alpha_i^2\Big)^{|D|}\,\mathrm{pdet}(H).$$

For an even split this factor is $m^{|S|-|D|}$. When $|S| = |D|$ it equals $\big(m\sum_i \alpha_i^2\big)^{|D|} \geq 1$, with equality iff the split is even.

The condition $|S| = |D|$ holds for a bias-free layer with as many inputs as outputs, as in the hidden layers of an equal-width MLP or the MLP block of a Transformer without biases. In general, the factor for an even split is $m^{|S|-|D|}$, so when a neuron has more duplicated than split weights, even splitting strictly increases the local volume. A bias is one such case: a post-activation split duplicates the bias along with the incoming weights, so $|D| = |S| + 1$ and the volume grows by a factor m. When H is singular, the factor $\det(K)$ is replaced by $\det(U^\top KU)$ with U an orthonormal basis for the range of H (Theorem H.3).

5 Creating Better-Conditioned Minima

Diagonal preconditioning has implications for optimization. The condition number $\kappa(H)$ governs standard worst-case convergence bounds for first-order methods. Classical results (Forsythe & Straus, 1955; Sluis, 1970) show that diagonal preconditioning improves κ when eigenvectors are not

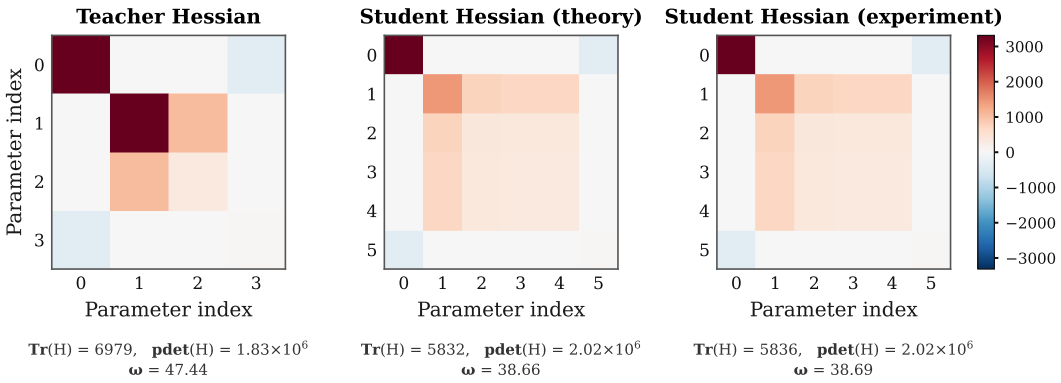


Figure 2: Theoretical and empirical Hessian comparison for a teacher-to-student expansion computed after optimizing to zero loss. *Left:* teacher Hessian. *Center:* student Hessian predicted by Theorem 4.3. *Right:* empirically measured student Hessian after training to zero loss. The theoretical prediction matches the experiment to within 0.03% in $\mathrm{Tr}(H)$, $\mathrm{pdet}(H)$, and ω.

element-wise equal in magnitude. While no guarantee of universal improvement exists, diagonal preconditioning is highly effective when eigenvectors are approximately *axis-aligned* (Figure 8). Empirically, we find Hessian eigenvectors of minima after optimization are more axis-aligned than random (Figure 9).

5.1 Trace and Average-Case Conditioning

The worst-case condition number $\kappa(H) = \lambda_{\max}/\lambda^{+}_{\min}$ is determined by only two eigenvalues and can be dominated by outliers. A more robust measure considers the full spectrum. The ratio of the arithmetic to geometric mean of the positive eigenvalues captures *average-case* conditioning and has been studied as an alternative to κ in the context of diagonal preconditioning and iterative methods (Doan & Wolkowicz, 2011; Jung et al., 2024; Ghadimi et al., 2025).

Theorem 5.1 (Trace Formula)**.** *Let H be the Hessian of the loss at a zero-loss minimum, and let $H' = B^\top H B$ be the Hessian at the image of that minimum under an m-way split of one neuron with coefficients $\alpha_1, \ldots, \alpha_m$ summing to 1. Let S be the split weights and D the duplicated weights. Then*

$$\mathrm{Tr}(H') = \mathrm{Tr}(H_{\setminus S,D}) + m \sum_{s\in S} H_{ss} + \Big(\sum_{i=1}^{m} \alpha_i^2\Big) \sum_{d\in D} H_{dd}, \tag{4}$$

where $H_{\setminus S,D}$ omits the rows and columns of $S \cup D$. Since $\sum_i \alpha_i^2 \geq 1/m$ with equality iff $\alpha_i = 1/m$, the even split minimizes $\mathrm{Tr}(H')$ at fixed m.

For the even split, treating the degree m as a continuous variable, the trace is a convex function of m. The new trace $\mathrm{Tr}(H')$ remains strictly smaller than the original trace $\mathrm{Tr}(H)$ for any degree $1 < m < \frac{\sum_{d\in D} H_{dd}}{\sum_{s\in S} H_{ss}}$, with the optimum m^* given in the proof.

Average-Case Condition Number. Using the AM-GM inequality for p positive eigenvalues, we see $\frac{1}{p}\sum_{i=1}^{p} \lambda_i \geq (\prod_{i=1}^{p} \lambda_i)^{1/p}$, with equality iff all λ_i are equal. The *average-case conditioning ratio* is:

$$\omega(H) = \frac{\frac{1}{p}\sum_{i=1}^{p} \lambda_i}{(\prod_{i=1}^{p} \lambda_i)^{1/p}} = \frac{\mathrm{Tr}(H)/p}{(\mathrm{pdet}\, H)^{1/p}}. \tag{5}$$

This ratio equals 1 when all eigenvalues are equal (perfect conditioning) and increases with eigenvalue spread. By Theorem 4.4, under even splits, the geometric mean $(\mathrm{pdet}\, H)^{1/p}$ is fixed when $|S| = |D|$ and increases in post-activation splits with biases. Since the denominator is fixed or increasing under such splits while the numerator (trace) can decrease, expanding width can create minima with better average-case conditioning than the corresponding minimum in the narrow network.

Average-Case Maximum Learning Rate. Paquette et al. (2022) show the average-case maximum learning rate goes as $\frac{2}{\mathrm{Tr}(H)}$. In conjunction with our results, this suggests wider networks contain minima equivalent to those of narrow networks that can be reached with larger learning rates.

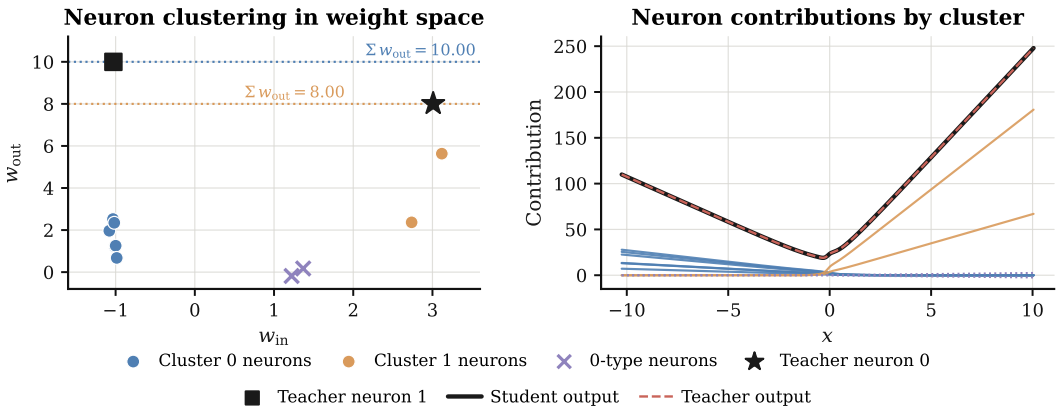


Figure 3: Neuron clustering at a global minimum of a width-10 student trained to match a width-2 teacher. *Left:* student weights in $(w_{\text{in}}, w_{\text{out}})$ space. Neurons cluster into two groups, each with the same w_{in} as the corresponding teacher neuron and outgoing weights summing to the teacher's value ($\Sigma = 10$ and $\Sigma = 8$). Two neurons are 0-type (output weights cancel). *Right:* per-cluster neuron contributions confirm each cluster implements one teacher neuron via splitting.

6 Probability of Initializing Near Minima

Theorem 6.1 (Volume Ratio)**.** *Let the teacher have minimal width n_f, scalar input and output, and a full-rank Hessian at θ^*. Initialize a width-n network uniformly on $[-b_n, b_n]$ with $b_n = 1/\sqrt{n}$ (Glorot & Bengio, 2010), and let P_n be the probability that this initialization lands in the ϵ-sublevel set of a global minimum. For ϵ small enough,*

$$\frac{P_{n_f+1}}{P_{n_f}} \;\geq\; c\,(n_f+1)^2\,\|w^{\text{out}}\|_1\,(\epsilon/C)^{1/4}, \tag{6}$$

where $\|w^{\text{out}}\|_1$ is the ℓ_1 norm of the teacher's outgoing weights, C bounds the fourth-order coefficient of the loss along the flat direction opened by a split, and c is an absolute constant.

This bound exceeds one once $\|w^{\text{out}}\|_1(n_f+1)^2 \gtrsim (C/\epsilon)^{1/4}$. The argument can be extended to multiple inputs and outputs and to Gaussian or He initialization, with only the constants changing (Theorem L.3).

7 Experiments

We test our predictions in three settings: small networks with guaranteed convergence to global minima, student-teacher experiments on MLPs, CNNs, and Transformers, and direct data fitting. The first setting tests the theory directly. The other two settings show that the same trends supported by the theory appear at practical scale, where we do not know that the trained solution is a symmetry expansion of a narrower one. We present these as complementary evidence.

7.1 Predictions at Zero Loss

We directly test Theorems 4.3 and 4.4 by training small networks to zero loss with L-BFGS. To rule out global minima outside the symmetry orbit of the teacher, we use the activation $\sigma_{\alpha,\gamma}(x) = \sigma_{\text{soft}}(x) + \alpha\,\sigma_{\text{sig}}(\gamma x)$ from Simsek et al. (2021), for which all global minima of the teacher-student loss are related by symmetry. Figure 2 shows the Hessian predicted by Theorem 4.3 matches the measured Hessian to within 0.03% in $\text{Tr}(H)$, $\text{pdet}(H)$, and ω, and the eigenvalues agree to within 0.2% (Figure 13). Post-optimization weights exhibit the predicted structure (Figure 3; see also Figure 12).

We perform three additional checks of the theory. First, the Hessian relation does not depend on the special activation: directly expanding trained sigmoid and ReLU teachers reproduces the predicted eigenvalues to relative error below 10^{-10}. For ReLU, where the teacher Hessian is singular, the range-restricted pseudo-determinant identity holds to the same precision (Table 1). Second, away

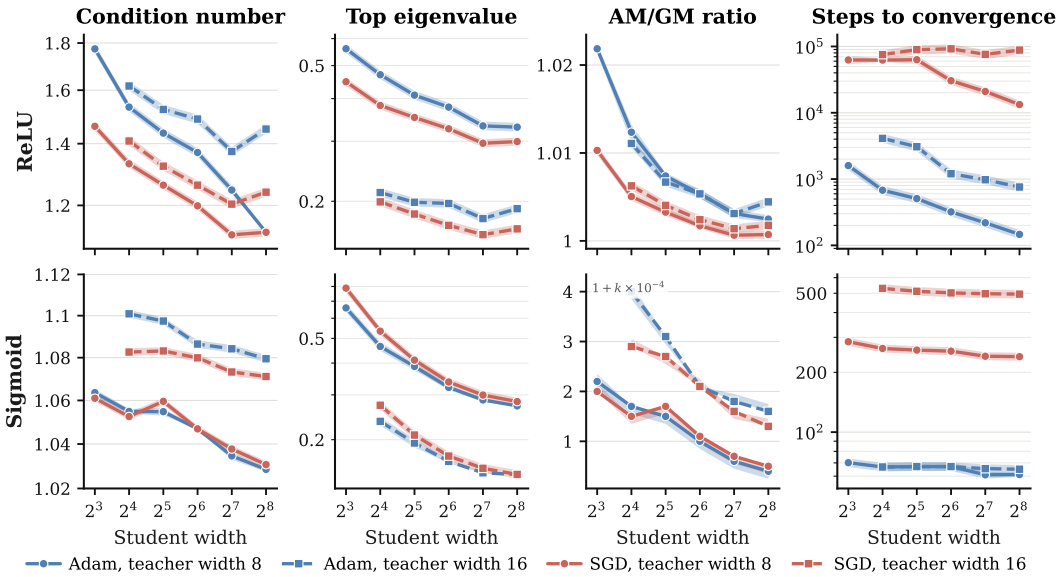


Figure 4: MLP student-teacher experiments (depth 3). Students use ReLU (top rows) and sigmoid (bottom rows), trained with Adam and SGD. Condition number, top eigenvalue, AM/GM ratio, and steps to convergence all improve with width and then saturate.

from zero loss, the Gauss-Newton residual is only a few percent or less of the Hessian at the losses reached in our MLP and CIFAR-10 experiments, and changes the condition number by less than one percent (Tables 2 and 3). Finally, for the volume bound, we verify the predicted quartic scaling along the flat direction, with fitted exponents between 3.99 and 4.01. The measured constant C places the crossover in Equation (6) at widths of a few to a few tens of neurons (Table 4).

7.2 Student-Teacher Experiments

These experiments ask whether the improvements in conditioning that follow from the theory also appear at practical scale and across architectures. We train student networks of increasing width and compute Hessian properties at convergence. Figure 4 shows MLP students trained with SGD and Adam: condition number, top eigenvalue, AM/GM ratio, and steps to convergence all improve with width and then saturate. The same trends hold across depths (Figure 17). CNN and Transformer results (Figure 15) show the same improvement.

7.3 Direct Data Fitting

To test whether the improvements depend on the teacher-student setup, we train networks directly on data. Figure 5 shows MLPs on California Housing regression and ConvNets on CIFAR-10 classification; the model cannot fully overfit in the latter setting. Conditioning improves with width in both cases, which suggests our finding may continue to be relevant in the underparameterized regime.

8 Discussion

8.1 Implications for Optimization

Our results characterize the minima that become available as width increases and their geometric properties. We do not determine which of these minima an optimizer will reach. There is, however, substantial evidence that optimization favors precisely the properties identified here. Larger-volume basins are reached more often from random initialization (Wu et al., 2017; Huang et al., 2020). At a fixed learning rate, gradient descent cannot settle at sufficiently sharp minima and instead stabilizes at flatter points (Cohen et al., 2021; Damian et al., 2023) and stochastic gradient noise further biases optimization away from sharp minima (Keskar et al., 2017; Zhu et al., 2019). Thus, although poorly conditioned or low-volume representatives may exist on the same symmetry orbit, there are several

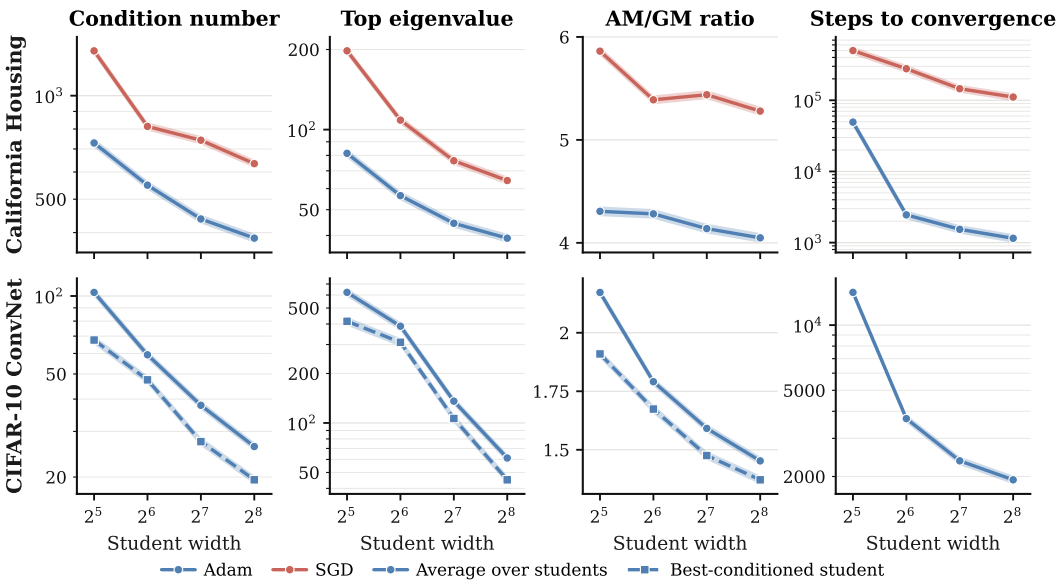


Figure 5: *Top:* MLPs trained directly on California Housing regression (no teacher). *Bottom:* ConvNets trained directly on CIFAR-10 classification (model cannot fully overfit). Conditioning improves with width in both settings.

mechanisms that favor the better-conditioned, higher-volume representatives made available by increasing width.

Importantly, this argument does not require the network to be overparameterized relative to the full learning problem. At any width n, a low-loss solution that can already be represented at some smaller width $n_f < n$ can be expanded to width n, introducing the symmetry degrees of freedom studied here. These degrees of freedom allow the same function to have better-conditioned and more numerous parameterizations at width n. In contrast, a solution whose minimal functional width is n has no such width redundancy. Thus, even in an underparameterized network, the mechanism can favor low-loss solutions that admit simpler, narrower representations.

8.2 Implications for Generalization

The argument above may also shed light on generalization. Consider noisy training data $\{x_i, y_i\}$ generated by a simple ground-truth function f, with $y_i = f(x_i) + \zeta_i$. The minimal functional width needed to represent f may be much smaller than that needed to fit the noise. At a fixed network width n, solutions representing f therefore have more width redundancy, and hence more symmetry-induced parameterizations, than overfitting solutions of larger minimal functional width. We conjecture that, among solutions with comparable loss, this gives simpler functions greater total volume and makes them easier to reach. In this sense, the optimization bias described in Section 8 may also act as a simplicity bias.

This gives a different interpretation of the old observation that generalizing minima tend to be wide (Hochreiter & Schmidhuber, 1997). Since symmetries can change sharpness without changing the function (Dinh et al., 2017), wideness itself cannot explain generalization. Rather, simplicity may explain both: simpler functions generalize better, and their greater width redundancy makes them more favorable to optimization. Perhaps the wonderful coincidence of deep learning is that these generalizing solutions also happen to be favored by our optimization process.

9 Conclusion

We study how the loss landscape changes as neurons are added, identifying two effects of width symmetries: diagonal Hessian preconditioning and increased volume near initialization. Our theoretical results are data and optimizer agnostic, depending only on the symmetry structure of the architecture, which prescribes the geometry of the loss landscape.

AI USE STATEMENT

We used generative AI tools to help polish the writing and prepare graphs and figures. We have reviewed all AI-assisted work. The authors are responsible for the final text, claims, graphs, and figures.

REPRODUCIBILITY STATEMENT

Full proofs of all theoretical results are given in Section H, and all experimental details, architectures and hyperparameters are listed in Section M. Code will be released before publication.

REFERENCES

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

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 242–252. PMLR, 2019.

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 244–253. PMLR, 2018.

Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Jeremy M. Cohen, Simran Kaur, Yuanzhi Li, J. Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2021.

Ronan Collobert. *Large Scale Machine Learning*. PhD thesis, Université de Paris VI, 2004.

Yaim Cooper. Global minima of overparameterized neural networks. *SIAM Journal on Mathematics of Data Science*, 3(2):676–691, 2021.

Alex Damian, Eshaan Nichani, and Jason D. Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. In *International Conference on Learning Representations*, 2023.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets, 2017.

X Doan and Henry Wolkowicz. Numerical computations and the $mega$-condition number. 01 2011.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A. Hamprecht. Essentially no barriers in neural network energy landscape. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1309–1318. PMLR, 2018.

Simon S. Du, Jason D. Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1675–1685. PMLR, 2019.

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.

G. E. Forsythe and E. G. Straus. On best conditioned matrices. *Proceedings of the American Mathematical Society*, 6(3):340–345, 1955.

K. Fukumizu and S. Amari. Local minima and plateaus in hierarchical structures of multilayer perceptrons. *Neural Networks*, 13(3):317–327, 2000. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(00)00009-5. URL https://www.sciencedirect.com/science/article/pii/S0893608000000095.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Saeed Ghadimi, Woosuk L. Jung, Arnesh Sujanani, David Torregrosa-Belén, and Henry Wolkowicz. New insights and algorithms for optimal diagonal preconditioning, 2025. URL https://arxiv.org/abs/2509.23439.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density, 2019.

Avrajit Ghosh, Soo Min Kwon, Rongrong Wang, Saiprasad Ravishankar, and Qing Qu. Learning dynamics of deep linear networks beyond the edge of stability, 2025.

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

J. Elisenda Grigsby, Kathryn Lindsey, and David Rolnick. Hidden symmetries of relu networks. In *International Conference on Machine Learning*, pp. 11734–11760. PMLR, 2023.

Charles Guille-Escuret, Hiroki Naganuma, Kilian Fatras, and Ioannis Mitliagkas. No wrong turns: The simple geometry of neural networks optimization paths, 2023.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997. doi: 10.1162/neco.1997.9.1.1.

Lei Huang, Xianglong Liu, Bo Lang, and Bo Li. Projection based weight normalization for deep neural networks, 2017.

W. Ronny Huang, Zeyad Emam, Micah Goldblum, Liam Fowl, J. K. Terry, Furong Huang, and Tom Goldstein. Understanding generalization through visualizations. *arXiv preprint arXiv:1906.03291*, 2020.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Amir Joudaki, Giulia Lanzillotta, Mohammad Samragh Razlighi, Iman Mirzadeh, Keivan Alizadeh, Thomas Hofmann, Mehrdad Farajtabar, and Fartash Faghri. Barriers for learning in an evolving world: Mathematical understanding of loss of plasticity, 2025. URL https://arxiv.org/abs/2510.00304.

Woosuk L. Jung, David Torregrosa-Belén, and Henry Wolkowicz. The ω-condition number: Applications to optimal preconditioning and low rank generalized jacobian updating, 2024. URL https://arxiv.org/abs/2308.13195.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

Daniel Kunin, Javier Sagastuy-Breña, Surya Ganguli, Daniel L.K. Yamins, and Hidenori Tanaka. Neural mechanics: Symmetry and broken conservation laws in deep learning dynamics. In *International Conference on Learning Representations*, 2021.

Flavio Martinelli, Berfin Şimşek, Wulfram Gerstner, and Johanni Brea. Expand-and-cluster: parameter recovery of neural networks. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Aaron Mishkin, Alberto Bietti, and Robert M. Gower. Level set teleportation: An optimization perspective, 2025.

Behnam Neyshabur, Ruslan Salakhutdinov, and Nathan Srebro. Path-SGD: Path-normalized optimization in deep neural networks, 2015.

Quynh Nguyen. On connected sublevel sets in deep learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4790–4799. PMLR, 2019.

Courtney Paquette, Elliot Paquette, Ben Adlam, and Jeffrey Pennington. Homogenization of sgd in high-dimensions: Exact dynamics and generalization properties, 2022. URL https://arxiv.org/abs/2205.07069.

Itay Safran and Ohad Shamir. On the quality of the initial basin in overspecified neural networks, 2015. URL https://arxiv.org/abs/1511.04210.

Levent Sagun, Utku Evci, V. Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks, 2017.

Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks, 2016.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, 2014.

Berfin Simsek, François Ged, Arthur Jacot, Francesco Spadaro, Clément Hongler, Wulfram Gerstner, and Johanni Brea. Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances, 2021.

A. van der Sluis. Condition numbers and equilibration of matrices. *Numerische Mathematik*, 14: 14–23, 1970. URL http://eudml.org/doc/131939.

Daniel Soudry and Elad Hoffer. Exponentially vanishing sub-optimal local minima in multilayer neural networks, 2017. URL https://arxiv.org/abs/1702.05777.

Héctor J Sussmann. Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural Networks*, 5(4):589–593, 1992.

Lei Wu, Zhanxing Zhu, and Weinan E. Towards understanding generalization of deep learning: Perspective of loss landscapes. *arXiv preprint arXiv:1706.10239*, 2017.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs V: Tuning large neural networks via zero-shot hyperparameter transfer, 2022.

Binchi Zhang, Zaiyi Zheng, Zhengzhang Chen, and Jundong Li. Beyond the permutation symmetry of transformers: The role of rotation for model fusion, 2025a.

Tianyue H. Zhang, Lucas Maes, Alan Milligan, Alexia Jolicoeur-Martineau, Ioannis Mitliagkas, Damien Scieur, Simon Lacoste-Julien, and Charles Guille-Escuret. Understanding Adam requires better rotation dependent assumptions, 2025b.

Zhengyan Zhang, Yixin Song, Guanghui Yu, Xu Han, Yankai Lin, Chaojun Xiao, Chenyang Song, Zhiyuan Liu, Zeyu Mi, and Maosong Sun. Relu^2 wins: Discovering efficient activation functions for sparse llms, 2024. URL https://arxiv.org/abs/2402.03804.

Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In *International Conference on Machine Learning*, 2019.

Liu Ziyin. Symmetry induces structure and constraint of learning, 2023. URL https://arxiv.org/abs/2309.16932.

Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Stochastic gradient descent optimizes over-parameterized deep ReLU networks, 2019.

CONTENTS

A Limitations and Future Work **15**

B Related Work **15**

C Zero-type Neurons **17**

D Per-Weight Splitting and the Inverse Transformation **18**

D.1 Per-Weight Coefficient Matrices 18

D.2 Row-wise Hessian Scaling 18

D.3 Inverse Transformation 18

E Which Activations Allow Pre-Activation Splitting? **19**

F The Overparameterization Groupoid **20**

G Transformation of Weight Space, Gradients and Hessian under Splitting **21**

H Proof Details **22**

H.1 Proof of Theorem 4.2 (Gradient Transformation) 22

H.2 Proof of Theorem 4.3 (Diagonal Preconditioning) 23

H.3 Proof of Theorem 4.4 and Associated Theorems 23

H.4 Proof of Theorem 5.1 (Trace Formula) 24

I Optimization Intuition via Rescale Symmetry **24**

J Axis-Aligned Eigenvectors and Transformer Rotation Symmetries **25**

K Local Volume of Minima **27**

L Derivation of the Volume Ratio **27**

L.1 Base network: P_n 27

L.2 Expanded network: P_{n+1} 28

L.3 The ratio P_{n+1}/P_n 29

M Experimental Appendix **29**

M.1 Architecture 29

M.2 Training 29

M.3 Compute Resources 31

M.4 Expansion Checks with Standard Activations 31

M.5 Size of the Gauss-Newton Correction 33

M.6 Calibrating the Volume Bound 34

M.7 Hessian Computation 34

M.8 Hyperparameters 35

N Notes on Plasticity Loss **35**

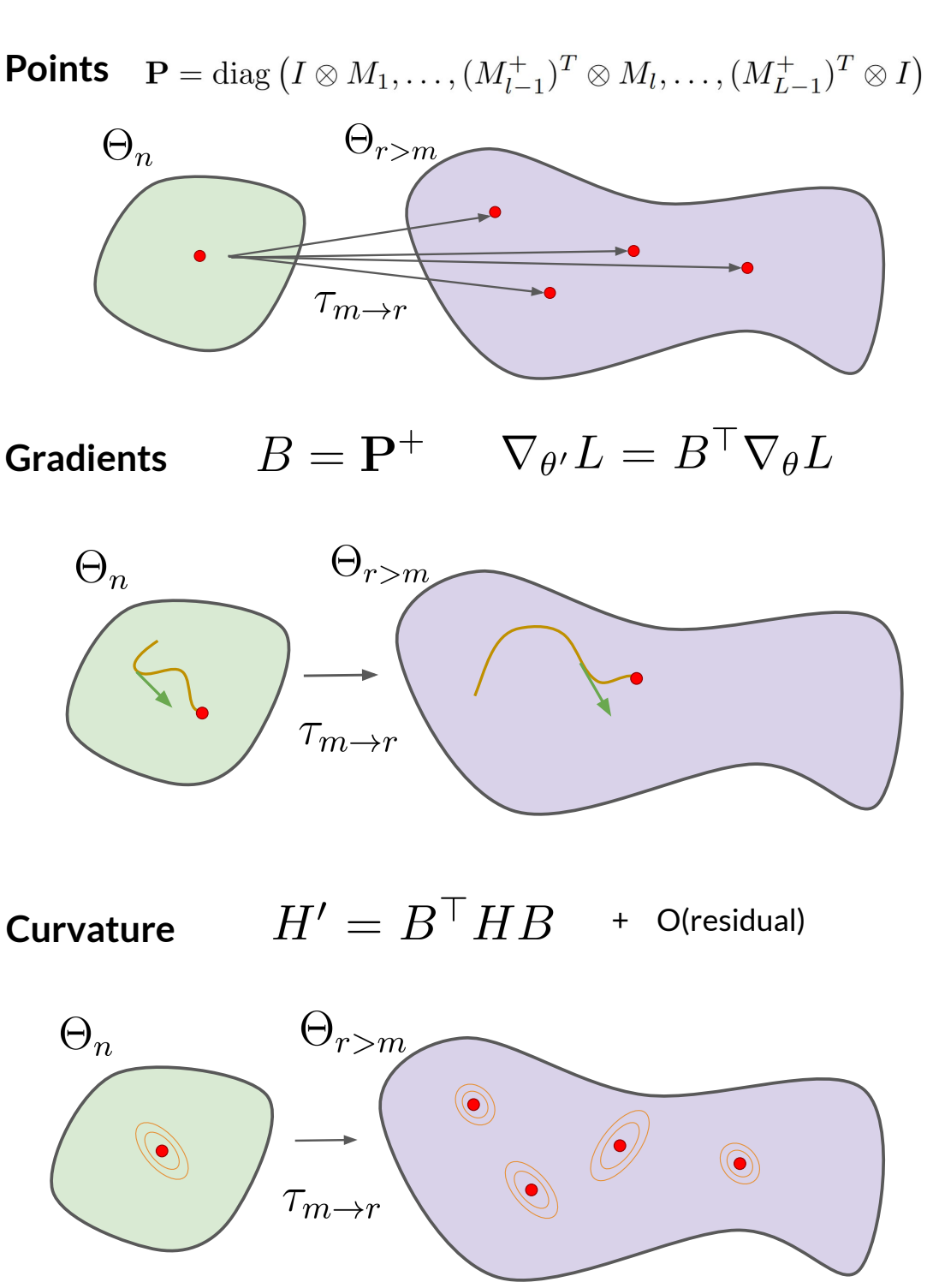


Figure 6: Overview of the results. How a splitting transformation $\tau_{m\to r}$ carries a narrow parameter space Θ_m into a wider one $\Theta_{r>m}$. *Top:* a single point maps to a whole family of equivalent points, one for each choice of split coefficients, given by $\mathbf{P}$. *Middle:* gradients are pulled back by $B^\top$, so a gradient step in the narrow network corresponds to a rescaled step in the wide one. *Bottom:* the Hessian at a zero-loss minimum maps to $B^\top H B$, and the level sets of the transported minima have different shapes depending on the split, which is the diagonal preconditioning of Theorem 4.3.

A Limitations and Future Work

Our theory is developed in a controlled setting, but several results suggest that the picture is more general. The relation $H' = B^\top HB$ is exact at zero loss, but we measure that it remains accurate beyond this setting. Moreover, the same improvement with width appears in direct data fitting and even in the mildly underparameterized regime (Section 7.3). Understanding when and why these conclusions continue to hold beyond the exact realizable setting is therefore an important direction for future work. Our analysis also focuses on width-wise symmetries and on the geometry of the resulting solution sets. Extending this picture to other forms of overparameterization, such as depth, and understanding the corresponding training dynamics are natural next steps.

B Related Work

A number of related works are cited in the main text. In addition, we found the following papers instructive in understanding symmetry, optimization, the loss landscape and overparameterization.

Geometry of overparameterization. Sussmann (1992) showed early on that for minimal-width networks, weights are determined up to permutation and sign-flip symmetries. Fukumizu & Amari (2000) later demonstrated that as width increases, the hierarchical structure of MLPs allows local minima to transform into saddle points, smoothing the landscape. Simsek et al. (2021) rigorously extended this to the overparameterized setting, identifying the continuous symmetries (neuron splitting and merging) that arise when width exceeds the functional minimum; their work provides the foundational groupoid structure for our analysis. Recently, Martinelli et al. (2024) leveraged these symmetries for parameter recovery, showing that overparameterized student networks naturally "cluster" their weights around the underlying minimal representation. We build upon these results by showing that these symmetries act as diagonal preconditioning on the Hessian and scale the probability mass of global minima.

Loss landscape structure. Cooper (2021) proves that for networks with d parameters and n data points ($d > n$), the global minima generically form a $(d-n)$-dimensional submanifold rather than isolated points. Nguyen (2019) shows that all sublevel sets of the loss are connected under mild overparameterization, ruling out "bad local valleys." Empirically, Garipov et al. (2018) and Draxler et al. (2018) independently found that minima from different training runs can be connected by paths of near-constant loss. Entezari et al. (2022) conjectured that after accounting for permutation symmetry, SGD solutions from different initializations lie in a single basin; Ainsworth et al. (2023) gave practical algorithms for aligning models modulo these symmetries. Our work adds to this picture by showing that overparameterization symmetries not only connect minima but reshape the curvature at those minima. Safran & Shamir (2015) bounded the probability of initializing in a basin with monotonic descent to a global minimum, and Soudry & Hoffer (2017) proved that sub-optimal local minima vanish exponentially in volume as width grows. Our Section 6 provides a specific symmetry-based mechanism for these basin-quality improvements.

Convergence theory. Jacot et al. (2018) introduced the neural tangent kernel (NTK), showing that in the infinite-width limit, training reduces to kernel regression with a fixed kernel. Building on this, several works proved that gradient descent on sufficiently wide networks converges to global minima at a linear rate: Du et al. (2019) for deep residual networks, Allen-Zhu et al. (2019) for general deep networks, and Zou et al. (2019) for SGD on deep ReLU networks. These results typically require the width to be polynomial in the number of samples and the depth. Chizat et al. (2019) showed that this favorable convergence arises from a "lazy training" regime where parameters stay near initialization and the model behaves as its linearization, which can underperform feature learning in practice. Our approach is different: rather than analyzing dynamics in a linearized regime, we characterize the geometric properties (conditioning, volume) of minima at finite width.

Implicit acceleration and deep linear networks. Deep linear networks, where $f(x) = W_L \cdots W_1 x$, are a standard testbed for optimization theory. Saxe et al. (2014) derived exact learning dynamics, showing that depth creates saddle-to-saddle transitions that progressively learn singular value components. Arora et al. (2018) showed that the resulting dynamics acts as implicit acceleration: the factored parameterization preconditions the optimization of the end-to-end matrix, with the effect depending on the singular values of the current solution. Ghosh et al. (2025) extend this analysis beyond the edge of stability, characterizing oscillatory dynamics at large learning rates. We extend

the preconditioning perspective in a different direction, from depth to width and from linear to general activations, showing that overparameterization symmetries provide diagonal preconditioning on the Hessian.

Conditioning and curvature. Sagun et al. (2017) showed empirically that overparameterized networks have Hessian spectra with a large bulk of near-zero eigenvalues and a few data-dependent outliers; the near-zero eigenvalues correspond to flat directions from overparameterization, while the outliers capture the curvature relevant to learning. Ghorbani et al. (2019) gave a more detailed eigenvalue density analysis, connecting spectral structure to properties of the data and architecture. Guille-Escuret et al. (2023) found that optimization paths in neural networks have surprisingly simple geometry despite the non-convexity of the loss. Yang et al. (2022) showed through μP that wider networks are more robust to learning rate choices, suggesting that width improves conditioning in a way that reduces hyperparameter sensitivity. Classical results on diagonal preconditioning (Forsythe & Straus, 1955; Collobert, 2004) show that coordinate rescaling can improve the condition number when eigenvectors are not uniformly distributed across coordinates; this underpins our analysis in Section 4.3.

Normalization and symmetry exploitation. Several practical methods explicitly navigate weight-space symmetries to find better-conditioned parameterizations. Weight normalization (Salimans & Kingma, 2016) decouples weight magnitude from direction, moving along the rescaling orbit. Path-SGD (Neyshabur et al., 2015) uses a path-regularized objective invariant to rescaling, so the trajectory is not distorted by the parameterization choice within an equivalence class. Mishkin et al. (2025) introduced level-set teleportation, which solves for a better-conditioned point on the same level set by optimizing over the symmetry group. All of these can be viewed as searching within a symmetry orbit for favorable parameterizations.

Why should symmetry help us think about width? Why not NTK instead? Many important symmetries of the network, such as permutation and rotation types, are also symmetries of the neural tangent kernel (Jacot et al., 2018). We therefore see the symmetry perspective as complementary to the NTK, and their synergy is an interesting direction. What the symmetry perspective adds is the following.

1. The mechanism gives a concrete path to understanding underparameterized networks. NTK-type results primarily apply to very-large-width regimes and do not clearly extend to underparameterized networks. By contrast, the approach here characterizes a general hierarchical structure in the loss landscape of neural networks of arbitrary width. It starts from a different and arguably less restrictive set of assumptions, which enables results potentially more relevant to practical optimization.
2. The view is constructive. We can identify exactly which set of weights represents the same function in a wider network, along with the full spectrum of those weights. This can be useful for applications: initialization when scaling models might benefit from a prior on where low-loss regions lie, obtained from narrower models that were trained, and there are practical implications for model merging and pruning.
3. It helps to more broadly understand the parameter-function map, which, despite being a central object in the theory of deep learning, remains poorly understood.
4. Overparameterization symmetries have far more degrees of freedom than the fixed-width symmetries usually studied. A rescaling symmetry contributes one degree of freedom per neuron, whereas the number of parameters grows quadratically with width. Splitting acts on every incoming and outgoing weight of a neuron, so the freedom it introduces scales with the size of the model rather than with the number of neurons, likely making it more important for optimization.

There are drawbacks to the approach, however. For example, it remains difficult to prove completeness of the symmetries for different architectures and activations.

C ZERO-TYPE NEURONS

The main text describes two function-preserving operations for arbitrary activations. The first (neuron duplication) is characterized in Section 3.3 using duplication matrices M and pseudoinverses M^+. Here we give an analogous characterization of the second: adding *0-type neurons*, whose total contribution to the network output cancels.

Setup. Consider a single hidden layer with n neurons, input weight matrix $W_l \in \mathbb{R}^{n \times d_{\text{in}}}$ and output weight matrix $W_{l+1} \in \mathbb{R}^{d_{\text{out}} \times n}$. We expand to $n + k$ neurons via expansion matrix $M_0 \in \mathbb{R}^{(n+k)\times n}$ and contraction matrix $M_0^+ \in \mathbb{R}^{n\times(n+k)}$:

$$M_0 = \begin{bmatrix} I_n \\ A \end{bmatrix}, \qquad M_0^+ = [I_n \quad Z], \tag{7}$$

where $A \in \mathbb{R}^{k\times n}$ and $Z \in \mathbb{R}^{n\times k}$. The transformation $W_l \to M_0 W_l$ and $W_{l+1} \to W_{l+1} M_0^+$ adds k new neurons with input weights AW_l and effective output weights $W_{l+1}Z$.

Function preservation. Verifying $f_\theta(x)$ is preserved:

$$\begin{aligned} W_{l+1} M_0^+ \sigma(M_0 W_l x) &= [W_{l+1} \quad W_{l+1}Z] \begin{bmatrix} \sigma(W_l x) \\ \sigma(AW_l x) \end{bmatrix} \\ &= W_{l+1}\,\sigma(W_l x) + W_{l+1} Z\,\sigma(AW_l x). \end{aligned} \tag{8}$$

The new neurons' total contribution $W_{l+1} Z\,\sigma(AW_l x)$ must vanish for all x. Partition the new neurons into groups by equality of their input weights (rows of AW_l). For arbitrary σ, this constraint is equivalent to: for each group G_g, the corresponding columns of $W_{l+1}Z$ sum to the zero vector. Writing $\mathbf{1}_g \in \mathbb{R}^k$ for the indicator vector of group G_g:

$$W_{l+1} Z\, \mathbf{1}_g = 0 \quad \text{for every group } g. \tag{9}$$

Two special cases.

- **Identically zero outputs:** when no two new neurons share an input weight (rows of AW_l are distinct), each group has size 1 and the constraint reduces to $W_{l+1}Z = \mathbf{0}$, i.e., every column of Z lies in the right null space of W_{l+1}.
- **Cancellation:** when new neurons come in pairs (or larger groups) with shared input weights, each group's columns of $W_{l+1}Z$ must sum to zero, but individual columns may be nonzero.

Relation to duplication. The 0-type expansion is complementary to neuron duplication. Duplication copies existing neurons and redistributes their output weights (the new neurons are functionally redundant copies of existing ones). The 0-type expansion adds genuinely new neurons whose total output cancels. Together, these two operations generate the full set of overparameterization symmetries for arbitrary activations described in Section 3.3.

Example: Adding 0-type Neurons

For a network with $n = 2$ hidden neurons and $d_{\text{out}} = 2$, adding two 0-type neurons sharing their input weight (rows 3 and 4 of M_0 are equal):

$$M_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ a_1 & a_2 \\ a_1 & a_2 \end{bmatrix}, \qquad M_0^+ = \begin{bmatrix} 1 & 0 & z_1 & -z_1 \\ 0 & 1 & z_2 & -z_2 \end{bmatrix}.$$

Both new neurons share the pre-activation $(a_1 W_l[1,:] + a_2 W_l[2,:])x$, and their output contributions $W_{l+1} z\,\sigma(\cdot)$ and $-W_{l+1} z\,\sigma(\cdot)$ cancel for any $z = (z_1, z_2)^\top$. The parameters a_1, a_2, z_1, z_2 are all free.

D Per-Weight Splitting and the Inverse Transformation

D.1 Per-Weight Coefficient Matrices

The formulation in Simsek et al. (2021) requires scaling the entire outgoing weight vector together. However, there exists a strictly richer symmetry: the splitting coefficients need not be shared across the entire outgoing weight vector.

Let $W_{l+1} \in \mathbb{R}^{d\times n}$ have columns $\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_n$. If column i is duplicated m_i times, we can choose a corresponding coefficient matrix $C_i \in \mathbb{R}^{d\times m_i}$ with normalized rows: $\sum_{j=1}^{m_i}(C_i)_{kj} = 1$ for all rows k.

The transformed output weights are:

$$W_{l+1} \to [C_1 \odot \vec{a}_1 \quad C_2 \odot \vec{a}_2 \ \ldots \ C_n \odot \vec{a}_n]\,, \tag{10}$$

where $\odot$ denotes element-wise multiplication and broadcasting applies column-wise (equivalently, the i-th block is given by $\mathrm{diag}(\vec{a}_i)C_i$).

Example: Per-Weight Splitting Coefficients

For $M = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$, we can transform a 2×2 matrix:

$$W_{l+1} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \to \begin{bmatrix} \mu_1 a_1 & (1-\mu_1)a_1 & a_2 \\ \mu_2 a_3 & (1-\mu_2)a_3 & a_4 \end{bmatrix}$$

for constants $\mu_1, \mu_2 \in \mathbb{R}$. Crucially, μ_1 and μ_2 can be chosen independently, giving per-weight control rather than per-neuron control.

D.2 Row-wise Hessian Scaling

The per-weight splitting above forces us to scale rows/columns in the Hessian as neuron-wise blocks. Here we discuss the gradient behavior when splits use individual weight-wise coefficients c_{jk}.

Theorem D.1. *Using the notation from Section 4.1, if our splits use individual coefficients per weight c_{jk}, we get:*

$$\frac{\partial L}{\partial w'^{(l)}_{ij}} = h_i^{(l-1)}(\sigma^l)' \sum_k c_{jk} w^l_{jk} \delta_k^{(l+1)} \tag{11}$$

$$= \frac{\sum_k c_{jk} w^l_{jk} \delta_k^{(l+1)}}{\sum_k w^l_{jk} \delta_k^{(l+1)}} \frac{\partial L}{\partial w^{(l)}_{ij}} \tag{12}$$

$$:= d_j \frac{\partial L}{\partial w^{(l)}_{ij}}. \tag{13}$$

The coefficient d_j only depends on index j, so we cannot control the gradient of individual weights independently.

D.3 Inverse Transformation

Given an expanded network with duplication matrix $M \in \mathbb{R}^{r\times n}$ and contraction M^+, the inverse transformation recovers the original width-n parameterization. For post-activation splitting, where neuron i was duplicated m_i times, the inverse groups each set of m_i copies back into a single neuron: the input weights are averaged across copies (recovering the original, since all copies are identical by construction), and the output weights are summed.

Concretely, the inverse uses the even-split pseudoinverse $M^+_{\text{even}} \in \mathbb{R}^{n\times r}$ for the input side and the original expansion matrix $M \in \mathbb{R}^{r\times n}$ for the output side:

$$W'_l \to M^+_{\text{even}} W'_l, \quad W'_{l+1} \to W'_{l+1} M, \tag{14}$$

where $(M^+_{\text{even}})_{i,j} = 1/m_i$ if column j is a copy of neuron i and 0 otherwise. The input averaging recovers the shared weights since $M^+_{\text{even}} M = I_n$. Right-multiplying by M naturally sums the split output weights: because the expanded weights are $W'_{l+1} = W_{l+1} M^+$, the inverse operation yields $W'_{l+1} M = W_{l+1} M^+ M = W_{l+1} I_n = W_{l+1}$.

For pre-activation splitting (ReLU), the roles reverse: output weights are averaged and input weights are summed to undo the splitting coefficients D_{m_i}.

Note that the inverse is only well-defined on the image of the forward transformation, i.e., on networks whose weights exhibit the tied structure produced by expansion. This is why the symmetries form a groupoid rather than a group (Section F).

Example: Inverse of Post-Activation Splitting

For $M = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$ (duplicating neuron 1), suppose the expanded network has:

$$W'_l = \begin{bmatrix} \vec{w}_1 \\ \vec{w}_1 \\ \vec{w}_2 \end{bmatrix}, \quad W'_{l+1} = \begin{bmatrix} \mu_1 a_1 & (1-\mu_1) a_1 & a_2 \\ \mu_2 a_3 & (1-\mu_2) a_3 & a_4 \end{bmatrix}.$$

The inverse uses $M^+_{\text{even}} = \left[\begin{smallmatrix} 1/2 & 1/2 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right]$ and the original matrix M:

$$M^+_{\text{even}} W'_l = \begin{bmatrix} \vec{w}_1 \\ \vec{w}_2 \end{bmatrix},$$

$$W'_{l+1} M = \begin{bmatrix} \mu_1 a_1 + (1-\mu_1) a_1 & a_2 \\ \mu_2 a_3 + (1-\mu_2) a_3 & a_4 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix},$$

recovering the original parameterization.

E Which Activations Allow Pre-Activation Splitting?

Example: Pre-Activation Splitting (ReLU)

Even split of input weights, duplicating outputs:

$$M' = \begin{bmatrix} \frac{1}{2} & 0 \\ \frac{1}{2} & 0 \\ 0 & 1 \end{bmatrix}, \quad M'^{+} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Input weights are halved; output weights are copied. This preserves the function because ReLU is equivariant to positive scaling.

Understanding general pre-activation splitting requires solving the functional equation representing a single split:

$$\forall x \in \mathbb{R}, \quad \sigma(\alpha x) + \sigma(\beta x) = \sigma(x), \tag{15}$$

for some expansion constants α, β. A direct observation is that setting $x = 0$ yields $2\sigma(0) = \sigma(0)$, implying $\sigma(0) = 0$. This immediately rules out activations with nonzero origins, such as the standard sigmoid.

To solve this systematically, we first restrict our attention to the domain $x > 0$ and assume $\alpha, \beta > 0$. We can map this to a continuous difference equation by substituting $x = e^t$ and defining $y(t) = \sigma(e^t)$. Letting $\alpha = e^a$ and $\beta = e^b$, the equation becomes:

$$y(t+a) + y(t+b) = y(t). \tag{16}$$

Applying the Laplace transform (assuming mild growth conditions on y) yields $(e^{as} + e^{bs} - 1)Y(s) = 0$. For non-trivial solutions ($Y(s) \neq 0$), s must be a root of the characteristic equation, meaning s lies in the set:

$$S_{a,b} = \{ s \in \mathbb{C} \mid e^{as} + e^{bs} = 1 \}. \tag{17}$$

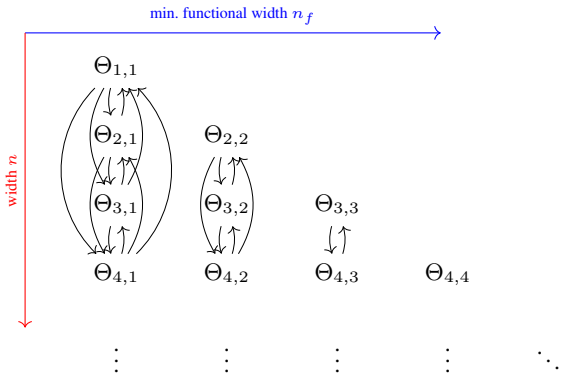


Figure 7: The overparameterization groupoid. Θ_{n,n_f} denotes weights of width n and minimum functional width n_f. Arrows indicate symmetry transformations that connect equivalent parameters across the widths.

If the ratio a/b is rational (i.e., $a/b = p/q$ with $\gcd(p, q) = 1$), the substitution $u = e^{s/q}$ reduces this to a polynomial equation $u^p + u^q = 1$, which has exactly $\max(p, q)$ roots for u by the Fundamental Theorem of Algebra. Taking the inverse Laplace transform for a given principal root yields base solutions of the form $y(t) = Ce^{st}$, which translates back to $\sigma(x) = Cx^s$, subject to the algebraic constraint $\alpha^s + \beta^s = 1$. When s is strictly real, these correspond to standard monomial functions.

However, we must also consider complex roots. When $s = \lambda + iw$, the solutions exhibit oscillatory behavior. For a purely imaginary root ($s = iw$, so $\lambda = 0$), the real-valued solutions take the form $\sigma(x) = \cos(w \ln x)$. Returning to the difference equation, this requires $\cos(wt + aw) + \cos(wt + bw) = \cos(wt)$. By angle addition, this holds when the phases form an equilateral triangle in the complex plane, yielding $(aw, bw) = (\pm\frac{\pi}{3} + 2\pi m, \mp\frac{\pi}{3} + 2\pi n)$ for $m, n \in \mathbb{Z}$. Because the limit $\lim_{x\to 0^+} \cos(w \ln x)$ is undefined, these purely oscillatory solutions are pathological and impractical for neural networks.

A careful treatment of signs maps our real roots ($s = 1, 2, \dots$) directly to practical piecewise-polynomial activations. This recovers both the standard ReLU ($s = 1$) and ReLU^2 ($s = 2$), which has recently emerged as a performant choice in sparse LLMs (Zhang et al., 2024).

F The Overparameterization Groupoid

The symmetry transformations form a *groupoid* rather than a group. A groupoid is a set of composable transformations following the group axioms that need not be defined on all objects. Here the objects are parameter vectors, and a transformation is a partial map from one parameter space to another. There are two reasons for using a groupoid.

Partial composition. A transformation $\tau_{a\to b} : \Theta_{a,n_f} \to \Theta_{b,n_f}$ can only compose with transformations originating from width b.

Restricted inverses. A contraction is only defined on parameters whose weights are tied by a previous expansion, that is, on the image of that expansion. In particular, we cannot contract below the minimal width needed to represent the function. Let $\mathcal{O}^L_{n_f}$ denote the groupoid of overparameterization operations on networks with L hidden layers and minimum functional width n_f. The key property is that the function is preserved across the orbit: $f_\theta = f_{\theta'}$ for any $\theta' \in \theta \cdot \mathcal{O}^L_{n_f}$.

Suppose $b > a$. The groupoid consists of actions for a single layer as the set of tuples $\tau_{a\to b} = \{(M_{a\to b}, M^+_{a\to b})\}$, where $M_{a\to b}$ is a valid expansion matrix and $M^+_{a\to b}$ a corresponding contraction (pseudoinverse) from Section 3. Figure 7 illustrates the structure: each column represents weights with a fixed minimum width n_f, and arrows connect equivalent parameterizations across different widths. The remainder of this section gives the full construction, including the per-weight coefficient form.

Let $\Gamma^{(L,N)}$ be the set of functions exactly representable by a network with L hidden layers and N neurons per layer, so that $\Gamma^{(L,N)} \subset \Gamma^{(L+p,N+q)}$ for any $p, q \in \mathbb{N}$. The objects of the groupoid are the sets

$$\Theta_{n,n_f} = \{\theta \text{ of width } n : f_\theta \in \Gamma^{(L,n_f)} \setminus \Gamma^{(L,n_f-1)}\}, \qquad n \geq n_f,$$

of width-n parameters whose function has minimal width exactly n_f. These are the nodes of Figure 7. Every morphism preserves the function and therefore the second index, so morphisms only run within a column of the figure.

For $b > a$, let $T_{a\to b}$ be the set of single-layer expansion operations,

$$T_{a\to b} = \{(M_{a\to b}, C_1, C_2 \dots C_a) \mid M_{a\to b} \text{ and } C_i \text{ are valid expansions from } a \text{ to } b\}, \tag{18}$$

where $M_{a\to b}$ and C_i are the expansion and coefficient matrices from Section 3 and Section D. Applied layer-wise, an element of $T^L_{a\to b} = (T_{a\to b})^L$ is a map $\Theta_{a,n_f} \to \Theta_{b,n_f}$ for every $n_f \leq a$.

Conversely, the contractions $T_{b\to a}$ are defined only on the image $\mathrm{im}(T_{a\to b}) \subset \Theta_{b,n_f}$. A contraction collapses $b - a$ duplicated pre-activation weights into a single weight and sums the corresponding output weights. Such duplicated weights exist for every $\theta \in \mathrm{im}(T_{a\to b})$ (see Appendix D.3), and since $a \geq n_f$ a contraction never leaves the column.

The overparameterization groupoid $\mathcal{O}^L_{n_f}$ has objects $\{\Theta_{n,n_f}\}_{n\geq n_f}$ and is generated by the layer-wise expansions, their contractions on the corresponding images, and the identities I_n:

$$\mathcal{O}^L_{n_f} = \langle T^L_{a\to b} \cup T^L_{b\to a} \cup \{I_n\} \mid b > a \geq n_f,\ n \geq n_f \rangle. \tag{19}$$

The partial binary operation is function composition, defined whenever the target of the first map is the source of the second, and it is associative. Every generator has an inverse on its image, so $\mathcal{O}^L_{n_f}$ satisfies the groupoid axioms. By construction, $f_\theta = f_{g\cdot\theta}$ whenever $g \cdot \theta$ is defined.

We remark that even some weight-space symmetries at a fixed width can be viewed as "local" or groupoid symmetries. Grigsby et al. (2023) provide a detailed analysis of such hidden symmetries in ReLU networks.

G Transformation of Weight Space, Gradients and Hessian under Splitting

For an L-layer neural network, assume we insert a transformation pair $M_l^+ M_l = I$ between every adjacent layer l and $l+1$. The forward parameters transform as follows:

- First layer: $W_1 \to M_1 W_1$
- Middle layers ($1 < l < L$): $W_l \to M_l W_l M_{l-1}^+$
- Final layer: $W_L \to W_L M_{L-1}^+$

By vectorizing the weights using the identity $\mathrm{vec}(ABC) = (C^T \otimes A)\mathrm{vec}(B)$, the parameter transformation matrix $\mathbf{P}$ across all layers is the block-diagonal matrix:

$$\mathbf{P} = \mathrm{diag}\left(I \otimes M_1, \dots, (M_{l-1}^+)^T \otimes M_l, \dots, (M_{L-1}^+)^T \otimes I\right) \tag{20}$$

Because parameters are contravariant while gradients are covariant, the gradient vector ∇L_θ transforms via the transpose of the pseudoinverse of the parameter transformation: $\nabla L_{\theta'} = (\mathbf{P}^+)^T \nabla L_\theta$.

Letting $B^T = (\mathbf{P}^+)^T$, the correct transformation for the expanded gradient vector across the entire network is $\nabla L_{\theta'} = B^T \nabla L_\theta$, where B^T is the block-diagonal matrix:

$$B^T = \mathrm{diag}\left(I \otimes (M_1^+)^T, \dots, M_{l-1} \otimes (M_l^+)^T, \dots, M_{L-1} \otimes I\right) \tag{21}$$

Hessian on the orbit. Write f_θ for the network output and $\mathcal{L}(\theta) = \mathbb{E}\,\ell(f_\theta(x), f^*(x))$. Differentiating twice,

$$H_\theta = \mathbb{E}\big[J_\theta^\top\, \ell''\, J_\theta\big] + \mathbb{E}\big[\ell'\, \nabla^2_\theta f_\theta\big], \qquad J_\theta = \frac{\partial f_\theta}{\partial \theta}. \tag{22}$$

The first term is the Gauss-Newton part. The second is proportional to the residual ℓ' and vanishes at a zero-loss minimum. On the orbit the expanded network computes the same function, and each expanded parameter enters the output through the original parameter it copies or splits, with the coefficients recorded in B, so $J_{\theta'} = J_\theta B$; this is Theorem 4.2 applied to the output rather than to the loss. The Gauss-Newton part therefore transforms as $B^\top \mathbb{E}[J_\theta^\top \ell'' J_\theta]\, B$, and at a zero-loss minimum

$$H_{\theta'} = B^\top H_\theta B. \tag{23}$$

Off zero loss the residual term does not take this form, so the relation is exact precisely when the Gauss-Newton approximation is exact. Section M.5 measures the size of the residual term at the losses reached in our experiments.

Block form for even splits. Partitioning the weights into O (other), S (split), and D (duplicated) with $|S| = |D|$, an order-m even split transforms the Hessian as

$$H = \begin{bmatrix} H_{OO} & H_{OS} & H_{OD} \\ H_{OS}^\top & H_{SS} & H_{SD} \\ H_{OD}^\top & H_{SD}^\top & H_{DD} \end{bmatrix} \;\to\; H' = \begin{bmatrix} H_{OO} & H'_{OS} & \frac{1}{m} H'_{OD} \\ H'^\top_{OS} & H'_{SS} & \frac{1}{m} H'_{SD} \\ \frac{1}{m} H'^\top_{OD} & \frac{1}{m} H'^\top_{SD} & \frac{1}{m^2} H'_{DD} \end{bmatrix}, \tag{24}$$

where $H'_{XX} = H_{XX} \otimes \mathbf{1}_{m\times m}$. Every duplicated row/column appears m times in its block; the $1/m$ and $1/m^2$ factors come from the $\alpha_i = 1/m$ row scalings of B for duplicated weights.

H Proof Details

H.1 Proof of Theorem 4.2 (Gradient Transformation)

We restate the per-weight form of Theorem 4.2 and prove it. The block form follows by vectorizing the expanded weights layer by layer.

Theorem H.1 (Theorem 4.2, per-weight form)**.** *Consider a splitting operation with coefficients $\alpha_1, \alpha_2, \ldots, \alpha_m$ summing to 1, where each coefficient multiplies the full vector of incoming or outgoing weights. Let D denote the set of duplicated weights. Then for the duplicated weight w'_i associated with coefficient α_i, $\frac{\partial L}{\partial w'_i} = \alpha_i \frac{\partial L}{\partial w}$. The gradients of all other weights, including the split weights, are unchanged.*

Corollary H.2 (Even Split)**.** *For an order-m even split (all $\alpha_i = 1/m$), the gradient of each duplicated weight is $\frac{1}{m}\frac{\partial L}{\partial w}$.*

Write $a^{(l)} = W_l h^{(l-1)}$ and $h^{(l)} = \sigma(a^{(l)})$ for the pre-activations and activations at layer l, and $\delta^{(l)} = \partial L / \partial a^{(l)}$ for the backpropagated error, so that

$$\delta^{(l-1)} = \sigma'(a^{(l-1)}) \odot (W_l)^\top \delta^{(l)}, \qquad \frac{\partial L}{\partial (W_l)_{ji}} = \delta_j^{(l)}\, h_i^{(l-1)}.$$

Post-activation splitting. Neuron j of layer l is copied m times and its outgoing weights are split, $(W_{l+1})_{kj} \to \alpha_i (W_{l+1})_{kj}$ for copy i. Each copy has the same activation $h_j^{(l)}$, and $\delta^{(l+1)}$ is unchanged because the function is unchanged, so the gradient of each split outgoing weight is $\delta_k^{(l+1)} h_j^{(l)}$, the same as before. The error reaching copy i is $\delta_{j,i}^{(l)} = \sigma'(a_j^{(l)}) \sum_k \alpha_i (W_{l+1})_{kj} \delta_k^{(l+1)} = \alpha_i \delta_j^{(l)}$, so the gradient of each duplicated incoming weight is α_i times the original.

Pre-activation splitting. The incoming weights are split, $(W_l)_{ji} \to \alpha_i (W_l)_{ji}$ for copy i, and the outgoing weights are copied. For a positively homogeneous activation, copy i has activation $\alpha_i h_j^{(l)}$ and σ' unchanged. The error reaching each copy is $\delta_j^{(l)}$, so the gradient of each split incoming weight is $\delta_j^{(l)} h_i^{(l-1)}$, unchanged, while the gradient of each duplicated outgoing weight is $\delta_k^{(l+1)} \alpha_i h_j^{(l)}$, which is α_i times the original.

Layers below the split are unaffected. With $W_l \to M W_l$ and $W_{l+1} \to W_{l+1} M^+$, the expanded pre-activation is $M a^{(l)}$ and, since σ acts pointwise and M only copies rows, $\sigma'(M a^{(l)}) = M \sigma'(a^{(l)})$.

Then

$$(MW_l)^\top \Big(M\sigma'(a^{(l)}) \odot (W_{l+1}M^+)^\top \delta^{(l+1)} \Big) = (W_l)^\top \Big(\sigma'(a^{(l)}) \odot M^\top (M^+)^\top (W_{l+1})^\top \delta^{(l+1)} \Big)$$
$$= (W_l)^\top \Big(\sigma'(a^{(l)}) \odot (W_{l+1})^\top \delta^{(l+1)} \Big),$$

using $M^\top (M^+)^\top = (M^+ M)^\top = I$. So $\delta^{(l-1)}$ and all gradients below layer l are unchanged. The same computation with the roles of M and M^+ exchanged covers pre-activation splitting.

H.2 Proof of Theorem 4.3 (Diagonal Preconditioning)

Applying Sylvester's identity and letting $K = BB^\top \in \mathbb{R}^{a\times a}$, the characteristic polynomial of H' is

$$\det(H' - \lambda I_b) = (-\lambda)^{b-a} \det(HK - \lambda I_a) = (-\lambda)^{b-a} \det(K^{1/2} H K^{1/2} - \lambda I_a) \tag{25}$$

by similarity, since K is positive definite for a valid expansion. This shows that H' has exactly $b-a$ zero eigenvalues, and its remaining a eigenvalues are identical to those of the $a \times a$ matrix HK. Nothing in this argument requires H to be full rank; if H is singular, some of these a eigenvalues are also zero.

H.3 Proof of Theorem 4.4 and Associated Theorems

To evaluate the change in the pseudo-determinant under expansion, we first explicitly characterize the diagonal matrix $K = BB^\top \in \mathbb{R}^{a\times a}$. As established in the main text, the expanded Hessian is $H' = B^\top H B$, and the nonzero eigenvalues of H' match those of HK.

The matrix $B \in \mathbb{R}^{a\times b}$ maps the gradients of the b expanded parameters back to the a original parameters. Because $K = BB^\top$, the diagonal entries of K correspond exactly to the inner products of the rows of B:

- **Untouched weights:** The gradient mapping is a standard basis vector. The inner product is 1, yielding a diagonal entry of 1.
- **Split weights ($s \in S$):** a weight whose split copies have values α_i times the original (with $\sum_i \alpha_i = 1$). By Theorem 4.2, each copy has *unchanged* gradient, so the row in B contains m ones, yielding a diagonal entry of m.
- **Duplicated weights ($d \in D$):** a weight whose copies are *identical* to the original, but whose copy gradients are scaled by the splitting proportions $\alpha_{d,i}$ (Theorem 4.2). The row in B contains the $\alpha_{d,i}$ values, yielding a diagonal entry of $\sum_{i=1}^m \alpha_{d,i}^2$.

Because K is a diagonal matrix, its determinant is simply the product of these diagonal entries. Assuming the expansion involves an equal number of split and duplicated weights ($|S| = |D|$), we can pair the scaling factors of each split weight with a duplicated weight. The determinant simplifies to:

$$\det(K) = 1^{|O|} \cdot \prod_{s\in S} m \cdot \prod_{d\in D} \left(\sum_{i=1}^m \alpha_{d,i}^2 \right) = \prod_{d\in D} \left(m \sum_{i=1}^m \alpha_{d,i}^2 \right)$$

More generally, the first form $\det(K) = m^{|S|} \prod_{d\in D} (\sum_i \alpha_{d,i}^2)$ gives $\det(K) = m^{|S|-|D|}$ under even splitting, so volume is preserved only when $|S| = |D|$. When $|S| < |D|$, for example when a post-activation split is applied to a neuron with more inputs than outputs, or when biases are duplicated, even splitting decreases pdet and so increases the local volume; when $|S| > |D|$ it decreases it. For ReLU networks both pre- and post-activation splits are available, so one can always choose the side with fewer weights to split and avoid losing volume.

Remark H.3 (Singular base Hessian). If H is singular, for instance because a ReLU network already has a rescaling symmetry at minimal width, the identity $\mathrm{pdet}(H') = \mathrm{pdet}(H)\det(K)$ no longer holds. Write $H = U\Lambda U^\top$ with $U \in \mathbb{R}^{a\times r}$ an orthonormal basis for the range of H and Λ the $r \times r$ diagonal matrix of nonzero eigenvalues. Applying Sylvester's identity to $H' = B^\top U \Lambda U^\top B$ gives

$$\mathrm{pdet}(H') = \mathrm{pdet}(H)\ \det(U^\top K U),$$

which reduces to $\mathrm{pdet}(H)\det(K)$ when $r = a$. The factor $\det(U^\top KU)$ need not equal one for an even split, so the geometric mean of the nonzero eigenvalues is not in general preserved when H is singular; Theorem 4.3 and the trace formula are unaffected. We verify this identity numerically in Section M.4.

Theorem H.4 (Theorem 4.4, restated)**.** *Let H be full rank and let H' be obtained by an m-way split of one neuron with coefficients $\alpha_1, \ldots, \alpha_m$ summing to one, with split weights S and duplicated weights D. Then $\mathrm{pdet}(H') = m^{|S|}(\sum_i \alpha_i^2)^{|D|}\,\mathrm{pdet}(H)$. For an even split the factor is $m^{|S|-|D|}$; when $|S| = |D|$ it is $(m\sum_i \alpha_i^2)^{|D|} \geq 1$, with equality iff the split is even.*

Proof. From the expression above, $\det(K) = m^{|S|}\prod_{d\in D}\sum_i \alpha_{d,i}^2$, and every duplicated weight of the split neuron carries the same coefficients, so $\det(K) = m^{|S|}(\sum_i \alpha_i^2)^{|D|}$; with $\mathrm{pdet}(H') = \mathrm{pdet}(H)\det(K)$ this is the identity. For an even split $\sum_i \alpha_i^2 = 1/m$, giving $m^{|S|-|D|}$. By Cauchy–Schwarz, $\sum_i \alpha_i^2 \geq \frac{1}{m}(\sum_i \alpha_i)^2 = 1/m$ with equality iff all α_i are equal, so when $|S| = |D|$ the factor $(m\sum_i \alpha_i^2)^{|D|}$ is at least one with equality iff the split is even. □

H.4 Proof of Theorem 5.1 (Trace Formula)

Since $H' = B^\top HB$ and $K = BB^\top$ is diagonal, $\mathrm{Tr}(H') = \mathrm{Tr}(KH) = \sum_j K_{jj}H_{jj}$. Reading off the diagonal of K from the proof of Theorem 4.4: entries of H_{OO} are unchanged, each split weight s contributes mH_{ss}, and each duplicated weight d contributes $(\sum_i \alpha_i^2)H_{dd}$. Summing gives the formula. By the RMS–AM inequality $\sum_i \alpha_i^2 \geq \frac{1}{m}(\sum_i \alpha_i)^2 = 1/m$, with equality iff all α_i are equal, so even splits minimize the trace at fixed m.

Optimal Splitting Degree. Treating the splitting degree m as continuous, the trace formula is convex in m; we call its minimizer (the value of m that yields the smallest trace under the fixed split structure S, D) the *optimal splitting degree*. Setting $\partial\,\mathrm{Tr}(H')/\partial m = 0$,

$$m^* = \sqrt{\frac{\sum_{d\in D} H_{dd}}{\sum_{s\in S} H_{ss}}}. \tag{26}$$

Substituting m^* back into the trace formula (or applying AM-GM) yields the minimum trace

$$\min \mathrm{Tr}(H') = \mathrm{Tr}(H_{\setminus S,D}) + 2\sqrt{\sum_{d\in D} H_{dd}\sum_{s\in S} H_{ss}}. \tag{27}$$

I Optimization Intuition via Rescale Symmetry

Before analyzing the overparameterization symmetries from the main text, we build intuition with the simpler rescale symmetry, which is present even in exactly-parameterized networks. The core argument carries over to the overparameterization case. It is well documented that rescale symmetries impact the curvature of minima and can be exploited to improve optimization (Huang et al., 2017; Salimans & Kingma, 2016; Neyshabur et al., 2015; Mishkin et al., 2025).

Consider a two-layer ReLU network $f(x; W, U) = U \cdot \mathrm{ReLU}(Wx)$. Since ReLU is equivariant to positive rescaling, the transformation $g_\lambda(W, U) = (\lambda W, \lambda^{-1}U)$ for $\lambda \in \mathbb{R}^+$ preserves the function. More generally, each hidden neuron can be rescaled independently.

Gradients under rescaling. At a rescaled point $(\lambda W, \lambda^{-1}U)$, the gradients transform as:

$$\frac{\partial L}{\partial W'} = \lambda^{-1}\frac{\partial L}{\partial W}, \qquad \frac{\partial L}{\partial U'} = \lambda\frac{\partial L}{\partial U}. \tag{28}$$

This follows from the chain rule: $W' = \lambda W$ so $\partial L/\partial W'_{ij} = (\partial W_{ij}/\partial W'_{ij}) \cdot \partial L/\partial W_{ij} = \lambda^{-1}\partial L/\partial W_{ij}$, and similarly for U. Rescaling thus redistributes gradient magnitudes between the input and output weights. When $\lambda > 1$, input weight gradients shrink while output weight gradients grow, and vice versa. This is analogous to the gradient scaling under splitting (Theorem 4.2), where the splitting coefficients play the role of λ.

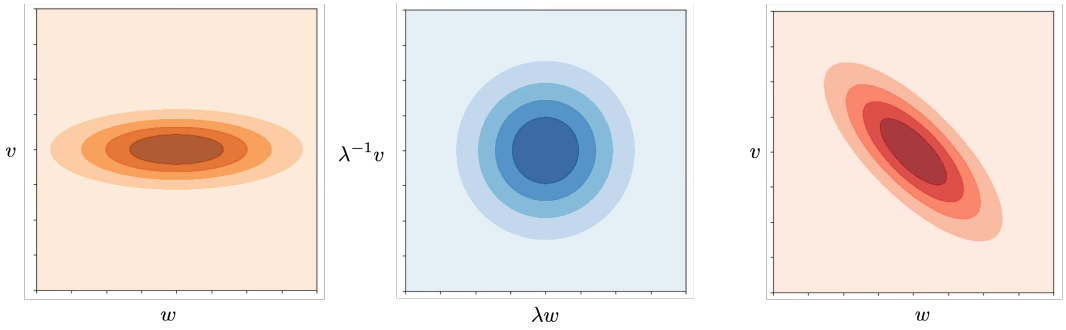


Figure 8: Diagonal preconditioning via symmetry transformations. Left: a poorly conditioned minimum (w^*, v^*) with elongated level sets. Center: rescaling to $(\lambda w^*, v^*/\lambda)$ compresses the w-axis and stretches the v-axis, producing rounder level sets that are easier to optimize. Right: when the principal curvature directions are not axis-aligned, diagonal rescaling cannot improve conditioning.

Hessian under rescaling. If (W^*, U^*) is a global minimum, rescaling gives a class of equivalent minima $[(W^*, U^*)]_{\mathcal{T}}$. The minimum (W^*, U^*) can be arbitrarily poorly conditioned, but the symmetry orbit contains solutions with different conditioning. The Hessian at (W^*, U^*) has the block structure:

$$H = \begin{bmatrix} H_{WW} & H_{WU} \\ H_{WU}^\top & H_{UU} \end{bmatrix}. \tag{29}$$

At the rescaled point $(\lambda W^*, \lambda^{-1} U^*)$, the Hessian becomes:

$$H(\lambda) = \begin{bmatrix} \lambda^{-2} H_{WW} & H_{WU} \\ H_{WU}^\top & \lambda^2 H_{UU} \end{bmatrix}. \tag{30}$$

This is diagonal preconditioning with $K = \mathrm{diag}(\lambda^{-2} I, \lambda^2 I)$, matching the structure from Section 4.3. Choosing λ lets us trade off curvature between the W and U blocks.

Limitations. While $\min_\lambda \kappa(H(\lambda)) \leq \kappa(H)$ always holds, rescaling cannot improve conditioning universally. To see this, consider the Rayleigh quotient for $x = (u, v)$, which is a weighted average of eigenvalues:

$$R(H(\lambda), x) = \frac{\lambda^{-2} u^\top H_{WW} u + 2 u^\top H_{WU} v + \lambda^2 v^\top H_{UU} v}{\|u\|^2 + \|v\|^2}. \tag{31}$$

The off-diagonal term $u^\top H_{WU} v$ is unaffected by λ, so if the extreme eigenvalues are dominated by the cross-block coupling, rescaling has little effect. Similarly, if both extreme eigenvalues come from a single block (say H_{WW}), rescaling scales them together and cannot change their ratio. Conditioning improves when the largest and smallest eigenvalues are split across the H_{WW} and H_{UU} blocks, which corresponds to the eigenvectors being approximately axis-aligned in the (W, U) decomposition. This is depicted in Figure 8.

J Axis-Aligned Eigenvectors and Transformer Rotation Symmetries

We empirically check the axis alignment of the Hessian eigenvectors of pretrained teacher networks across architectures.

In Figure 9, we compute the top-1000 Hessian eigenpairs via Lanczos at the trained parameters, and for each eigenvector v measure the participation ratio $\mathrm{PR}(v) = 1/\sum_i v_i^4$ on the parameter basis. The fourth power simply makes PR an interpretable count: if v has equal mass on k coordinates and zero on the rest, then $\sum_i v_i^4 = 1/k$, so $\mathrm{PR}(v) = k$. A lower power would not give this property: $\sum_i v_i^2 = 1$ for any unit v, so it cannot distinguish a one-hot eigenvector from a fully spread one. We

report $\mathrm{PR}(v)/n_{\mathrm{params}}$, the fraction of parameters the eigenvector effectively occupies; a uniformly random unit vector in $\mathbb{R}^{n_{\mathrm{params}}}$ has expected fraction $\approx 1/3$.

Across the three architectures we tested (a CNN trained on CIFAR-10, an absolute-position character Transformer pretrained on Alice in Wonderland, and the same Transformer with RoPE positional encoding) the Hessian eigenvectors lie below this random baseline (Figure 9), indicating a degree of axis alignment relative to random directions. The localization is strongest at the largest eigenvalues and decays toward the random baseline as one moves down the spectrum.

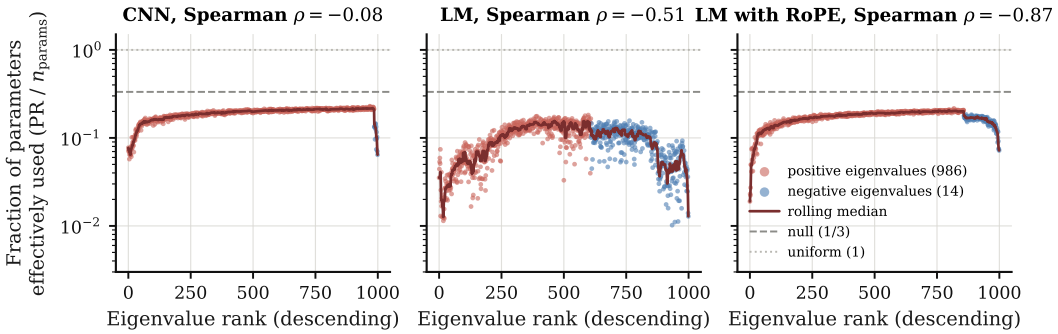


Figure 9: Fraction of parameters effectively used by each of the top-1000 Hessian eigenvectors of three pretrained teachers, against a random-unit-vector null ($\approx 1/3$, gray band). Red points correspond to positive eigenvalues, blue to negative; the dark red line is a rolling median.

To see where this localization sits in parameter space, we decompose each top eigenvector v into per-tensor energy fractions $\sum_{i\in T} v_i^2$ for each parameter tensor T, and divide by the parameter-count fraction $|T|/n_{\mathrm{params}}$ that a uniform random direction would assign to T. The resulting quantity (one per (eigenvector, tensor) pair) takes value 1 for a uniform direction and grows when the eigenvector concentrates more energy on a tensor than its size alone would predict. Figure 10 shows this decomposition for the top-20 eigenvectors of the RoPE Transformer teacher. The brightest columns correspond to the per-block additive biases on the residual stream (the attention output biases `attn.out.b` and the FFN output biases `ff.2.b` of every layer), each of which contains only 32 of the ∼56k model parameters but absorbs a disproportionate share of each top eigenvector's energy.

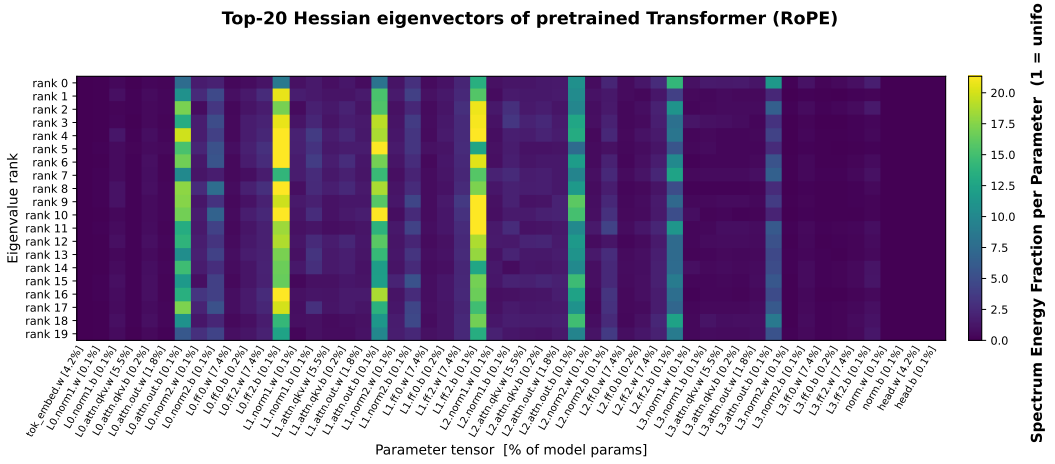


Figure 10: Spectrum energy fraction per parameter for the top-20 Hessian eigenvectors of the RoPE Transformer teacher, broken down by parameter tensor. Each column is a parameter tensor (annotated with its share of total parameters); each row is an eigenvector ranked by descending eigenvalue. Brighter cells indicate more eigenvector energy per parameter than a uniform random direction would assign.

Transformers have an additional rotation symmetry in their MLP weights, partially described in Zhang et al. (2025a) for networks without skip connections. In modern pre-norm Transformers with

skip connections, the symmetry is more restricted. In general, the symmetry performs a rotation in the MLP weight matrices $W \to R_{\text{left}} W R_{\text{right}}^\top$ for some orthogonal matrices $R_{\text{left}}, R_{\text{right}} \in O(n)$, acting jointly across adjacent blocks to preserve the residual stream.

To see how this rotates weight space, we apply the Kronecker product identity:

$$\text{vec}(R_{\text{left}} W R_{\text{right}}^\top) = (R_{\text{left}} \otimes R_{\text{right}})\text{vec}(W). \tag{32}$$

Intuitively, this rotates in unison the column weights by R_{left} and the row weights by R_{right}.

This rotation symmetry can aid optimization algorithms with built-in diagonal preconditioners, such as Adam. Recent work has noted the underperformance of Adam when rotating the preconditioner in Transformers but not ResNets (which lack this rotation symmetry), suggesting the minima Adam converges to typically have axis-aligned eigenvectors (Zhang et al., 2025b).

K Local Volume of Minima

We compute the volume of the local ϵ-sublevel set used in Section 4.4. Taylor-expanding the loss near a minimum θ^*, the local ellipsoidal region $\mathcal{E}_\epsilon(\theta^*)$ where the loss increases by at most ϵ satisfies $\frac{1}{2}(\theta - \theta^*)^\top H(\theta - \theta^*) \le \epsilon$.

Assuming the parameter space has dimension n, the affine change of variables $u = \frac{1}{\sqrt{2\epsilon}} H^{1/2}(\theta - \theta^*)$ maps the region to the unit n-ball $u^\top u \le 1$, with Jacobian determinant $(2\epsilon)^{n/2}(\det(H))^{-1/2}$. Integrating over the unit ball gives

$$\text{Vol}(\mathcal{E}_\epsilon(\theta^*)) = \frac{(2\epsilon)^{n/2}\ \text{Vol}(B_n)}{\sqrt{\det(H)}}, \tag{33}$$

where $\text{Vol}(B_n) = \pi^{n/2}/\Gamma(n/2+1)$. When H is singular (as for H' obtained by splitting), $\det(H)$ is replaced by the pseudo-determinant $\text{pdet}(H)$, giving the volume modulo flat symmetry directions used in Theorem 4.4.

L Derivation of the Volume Ratio

We derive the probability ratio when expanding from n to $n+1$ neurons (i.e. $k=1$). Throughout, the base network is at its minimal width, $n = n_f$, so that its global minima are exactly the $n!$ permutation copies of θ^* and its Hessian there is full rank. We fix $d_{\text{in}} = d_{\text{out}} = 1$, so the base network has $2n$ parameters and the expanded network has $2(n+1)$ parameters. The m-neuron network is initialized uniformly on $[-b_m, b_m]^{2m}$ where $b_m = 1/\sqrt{m}$.

Domain assumption. We restrict attention to base minima θ^* whose weights all satisfy $|w_j^{\text{in}}|, |w_j^{\text{out}}| \le b_{n+1}$ for every j. Under this condition, for any $\alpha \in [0,1]$ the split minimum $\theta^*_{\alpha,j}$ lies entirely inside $[-b_{n+1}, b_{n+1}]^{2(n+1)}$: the shared incoming weights are bounded by b_{n+1} by assumption, and $|\alpha w_j^{\text{out}}|, |(1-\alpha) w_j^{\text{out}}| \le |w_j^{\text{out}}| \le b_{n+1}$. The entire interval $\alpha \in [0,1]$ is therefore admissible.

L.1 Base network: P_n

Volume of the ϵ-sublevel ellipsoid. Fix a global minimum θ^*. The quadratic approximation of the loss defines an ellipsoid $\{\theta : \frac{1}{2}(\theta - \theta^*)^\top H(\theta - \theta^*) \le \epsilon\}$ in $\mathbb{R}^{2n}$. Assuming H is full rank (no flat directions for the base network), the ellipsoid has volume

$$\text{Vol}(\mathcal{E}_\epsilon(\theta^*)) \ = \ \frac{(2\epsilon)^n\ \text{Vol}(B_{2n})}{\sqrt{\det(H)}}, \tag{34}$$

where $\text{Vol}(B_{2n}) = \pi^n/\Gamma(n+1)$.

Summing over permutation copies. Permuting the n neurons gives $n!$ equivalent minima. Assuming the ϵ-sublevel sets around distinct permutation copies are disjoint (which holds when ϵ is small relative to the separation between minima), and that each ellipsoid lies inside $[-b_n, b_n]^{2n}$ (which holds when ϵ is small relative to the distance from θ^* to the boundary), the probability of initializing within loss ϵ of any global minimum of this type is

$$\boxed{P_n \;=\; \frac{n!\;(2\epsilon)^n\;\mathrm{Vol}(B_{2n})}{(2b_n)^{2n}\,\sqrt{\det(H)}}.} \tag{35}$$

L.2 Expanded network: P_{n+1}

Structure of minima. For the $k = 1$ expansion, a global minimum of the expanded network arises by splitting exactly one neuron $j \in \{1, \dots, n\}$ into two copies with outgoing weights $\alpha w_j^{\mathrm{out}}$ and $(1-\alpha)w_j^{\mathrm{out}}$ for some $\alpha \in [0, 1]$, while both copies share the incoming weight w_j^{in}. The corresponding parameter vector is

$$\theta^*_{\alpha,j} \;=\; \big(\dots,\; w_j^{\mathrm{in}},\, \alpha w_j^{\mathrm{out}},\; w_j^{\mathrm{in}},\, (1-\alpha)w_j^{\mathrm{out}},\; \dots\big) \;\in\; \mathbb{R}^{2(n+1)}.$$

Pseudo-determinant of the expanded Hessian. By Theorem 4.4 applied to the single split, the expanded Hessian H' has $2n$ nonzero eigenvalues and 2 zero eigenvalues. Its pseudo-determinant satisfies

$$\mathrm{pdet}(H') \;=\; \det(H)\;\cdot\; 2\big(\alpha^2 + (1-\alpha)^2\big). \tag{36}$$

Flat directions. The 2 zero eigenvalues correspond to two distinct directions.

- *Globally flat direction*: shifting $\alpha \mapsto \alpha + t$ (redistributing the outgoing weight) leaves the network function unchanged. We integrate over this direction explicitly below.
- *Locally flat direction*: the null direction of the Hessian block for the two copies' incoming weights is $v_\alpha = (1-\alpha, -\alpha)$, since linearizing the network output in incoming-weight perturbations $(\delta w_{\mathrm{in}}^{(1)}, \delta w_{\mathrm{in}}^{(2)})$ gives a change proportional to $\alpha\,\delta w_{\mathrm{in}}^{(1)} + (1-\alpha)\,\delta w_{\mathrm{in}}^{(2)}$. Consider a perturbation size s along v_α giving a set of parameters θ_s. Along this direction the first-order change in f_{θ_s} is 0, and since the loss is quadratic in f_{θ_s} it grows at most quartically (as s^4), so the sublevel set $\{L \le \epsilon\}$ has width $O(\epsilon^{1/4})$, contributing a factor $(\epsilon/C)^{1/4}$ for a constant $C > 0$ taken as a uniform upper bound on the fourth-order coefficient over $\alpha \in [0, 1]$ and admissible teacher minima.

Volume in the nonflat subspace. In the $2n$-dimensional subspace orthogonal to both flat directions:

$$\mathrm{Vol}\big(\mathcal{E}_\epsilon(\theta^*_{\alpha,j})^\perp\big) \;=\; \frac{(2\epsilon)^n\;\mathrm{Vol}(B_{2n})}{\sqrt{\mathrm{pdet}(H')}} \;=\; \frac{(2\epsilon)^n\;\mathrm{Vol}(B_{2n})}{\sqrt{\det(H)}\cdot\sqrt{2(\alpha^2+(1-\alpha)^2)}}. \tag{37}$$

Integrating over the globally flat direction. The orbit is the line segment $\theta^*_\alpha = (\dots, w_j^{\mathrm{in}}, \alpha w_j^{\mathrm{out}}, w_j^{\mathrm{in}}, (1-\alpha)w_j^{\mathrm{out}}, \dots)$ with tangent vector $\partial_\alpha \theta^*_\alpha$ of norm $\sqrt{2}\,|w_j^{\mathrm{out}}|$, so the arc-length element is $d\ell = \sqrt{2}\,|w_j^{\mathrm{out}}|\,d\alpha$. By the domain assumption, the full interval $\alpha \in [0, 1]$ is admissible. Integrating the per-α cross-section with respect to arc length,

$$I(1) \;:=\; \int_0^1 \frac{\sqrt{2}\,|w_j^{\mathrm{out}}|\,d\alpha}{\sqrt{2\big(\alpha^2+(1-\alpha)^2\big)}} \;=\; \sqrt{2}\,|w_j^{\mathrm{out}}|\,\ln(1+\sqrt{2}), \tag{38}$$

where the closed form follows by the substitution $u = \alpha - \frac{1}{2}$ and the standard arsinh integral. The two permutation copies $(\alpha, 1-\alpha)$ and $(1-\alpha, \alpha)$ are already counted in the global $(n+1)!$ permutation factor, so we divide by $2! = 2$.

Summing over choices of split neuron. There are n choices of which neuron j to split, each contributing $I(1) = \sqrt{2}\,|w_j^{\mathrm{out}}|\,\ln(1+\sqrt{2})$ to the orbit integral. Summing over j gives a total of

$\sqrt{2}\,\ln(1+\sqrt{2})\sum_{j=1}^{n}|w_j^{\text{out}}| = \sqrt{2}\,\ln(1+\sqrt{2})\,\|w^{\text{out}}\|_1$. Collecting all factors:

$$P_{n+1} \geq \underbrace{(2b_{n+1})^{-2(n+1)}}_{\text{init density}} \cdot \underbrace{(n+1)!}_{\text{perm. copies}} \cdot \underbrace{(\epsilon/C)^{1/4}}_{\text{locally flat direction}} \cdot \underbrace{\frac{(2\epsilon)^n\ \text{Vol}(B_{2n})}{\sqrt{\det(H)}}}_{\text{ellipsoid vol.}} \cdot \underbrace{\frac{\sqrt{2}\,\ln(1+\sqrt{2})\,\|w^{\text{out}}\|_1}{2}}_{\text{split neuron choices}}. \tag{39}$$

L.3 The ratio P_{n+1}/P_n

Dividing (39) by (35), and substituting Xavier initialization $b_m = 1/\sqrt{m}$ (Glorot & Bengio, 2010):

$$\frac{P_{n+1}}{P_n} \geq \underbrace{\frac{(2b_n)^{2n}}{(2b_{n+1})^{2(n+1)}}}_{=\frac{(n+1)^{n+1}}{4\,n^n}} \cdot \underbrace{\frac{(n+1)!}{n!}}_{=n+1} \cdot (\epsilon/C)^{1/4} \cdot \frac{\sqrt{2}\,\ln(1+\sqrt{2})\,\|w^{\text{out}}\|_1}{2}$$

$$= \frac{\sqrt{2}\,\ln(1+\sqrt{2})}{8} \cdot \|w^{\text{out}}\|_1\,(n+1)^2\left(\frac{n+1}{n}\right)^n (\epsilon/C)^{1/4}. \tag{40}$$

Since $((n+1)/n)^n \in [2, e)$ for all $n \geq 1$, dropping it together with the numerical prefactor gives the form stated in Equation (6).

Remark L.1 (Growth threshold). All factors in (40) are positive, so $P_{n+1}/P_n > 0$ always. The ratio exceeds 1 if and only if

$$\|w^{\text{out}}\|_1\,(n+1)^2\left(\frac{n+1}{n}\right)^n > \frac{8C^{1/4}}{\sqrt{2}\,\ln(1+\sqrt{2})\,\epsilon^{1/4}}. \tag{41}$$

Since $((n+1)/n)^n$ is increasing in n with $((n+1)/n)^n \geq 2$ for all $n \geq 1$, a sufficient condition is

$$\|w^{\text{out}}\|_1\,(n+1)^2 > \frac{8C^{1/4}}{\sqrt{2}\,\ln(1+\sqrt{2})\,\epsilon^{1/4}}. \tag{42}$$

Remark L.2 (Lower bound interpretation). The formula (40) counts only global minima arising from single-neuron splitting (the $k = 1$ duplication symmetry). The true P_{n+1} also receives contributions from other global minima not of this form, so (40) provides a lower bound on the true ratio.

Remark L.3 (Extensions). We wrote the derivation for the simplest case. None of the steps depend on it. Disjointness of the sublevel sets and the domain assumption both follow once ϵ is small enough. He initialization replaces $b_m = 1/\sqrt{m}$ by $\sqrt{2/m}$ and changes the bound by a constant factor. Gaussian initialization removes the domain assumption; the constant density $1/(2b_m)^{2m}$ is replaced by the minimum of the Gaussian density over each sublevel set, which gives a less compact but qualitatively identical bound that can be evaluated numerically. For p inputs and q outputs the parameter count $2n$ becomes $n(p+q)$ and the geometric factors become dimension dependent, which changes the constants and the exponent of ϵ but not the form of the argument.

M Experimental Appendix

M.1 Architecture

For the depth-1 sweep, networks are single-hidden-layer MLPs ($L = 1$) with input and output dimension equal to the teacher width ($d_{\text{in}} = d_{\text{out}} = 8$). Deeper sweeps and other experiment families use the architectures listed in Table 5. The teacher has hidden width $n_f = 8$; student widths are $n \in \{8, 16, 32, 64\}$, corresponding to overparameterization ratios $n/n_f \in \{1, 2, 4, 8\}$. We test both ReLU and sigmoid activations.

M.2 Training

Synthetic MLP student-teacher. For each configuration, we train 5 independent student networks in parallel to match the frozen teacher's outputs. At each training step, a fresh batch of inputs is

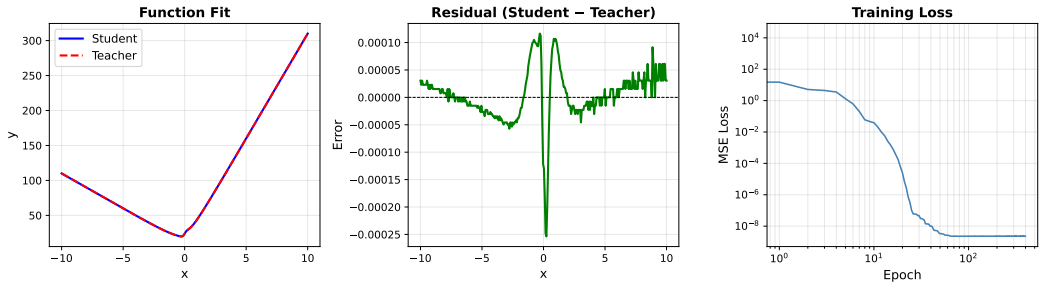


Figure 11: A width-3 student trained to match a width-2 teacher to zero loss via L-BFGS. *Left:* Student aligns exactly with teacher. *Center:* There is a small residual error. *Right:* Loss curve using L-BFGS.

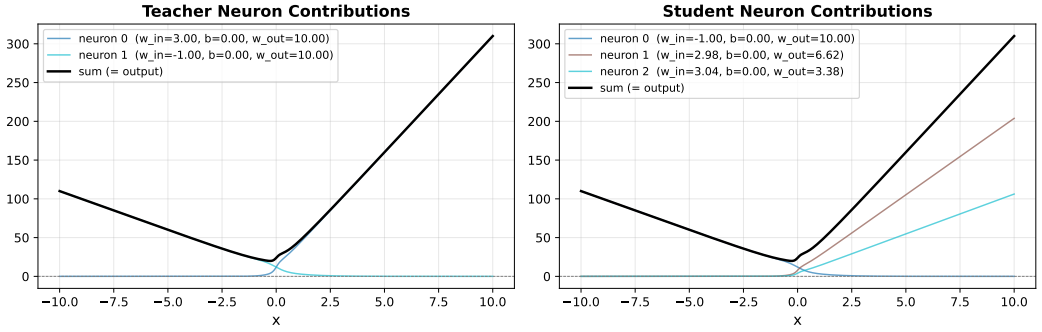


Figure 12: Weight structure at a global minimum of a width-3 student trained to match a width-2 teacher (zero loss via L-BFGS). *Left:* teacher neuron contributions. *Right:* student neuron contributions. Student neuron 0 replicates teacher neuron 1 exactly ($w_{\text{in}} = -1.00$), while neurons 1 and 2 split teacher neuron 0 ($w_{\text{in}} \approx 3.00$) with outgoing weights $6.62 + 3.38 = 10.00$, matching the predicted splitting structure.

sampled from $\mathcal{N}(0, I)$ and the teacher generates targets; each student sees the same batch. The loss is the mean squared error between student and teacher outputs.

We use maximal update parameterization μP (Yang et al., 2022) for width-consistent learning dynamics, with base width 8. We test two optimizers:

- **SGD**: learning rate 0.1, maximum 50,000 epochs.
- **Adam**: learning rate 0.01, maximum 5,000 epochs.

Training terminates early if the loss falls below 5×10^{-5} or if no improvement is observed for 20,000 consecutive epochs.

MLP on California Housing. As a no-teacher counterpart to the synthetic experiment, students are trained directly to predict median house value from the 8 standardized California Housing features, on a fixed 200-sample training subset. All other hyperparameters — μP, optimizer pair, learning rates, epoch budgets, early stopping — match the synthetic experiment.

CNN on CIFAR-10. The architecture is a 4-block ConvNet (Conv $\to$ BN $\to$ ReLU, with 2×2 MaxPool every two blocks) followed by global average pooling and a linear head. Channel widths $\{32, 64, 128, 256\}$ are swept. Two settings share hyperparameters except for the loss: *teacher-student*, matching the logits of a pretrained width-32 teacher under MSE (early-stop 5×10^{-3}), and *direct*, predicting CIFAR-10 labels under cross-entropy (early-stop 0.4). Both use Adam at 10^{-3}, batch size 256, maximum 5,000 epochs.

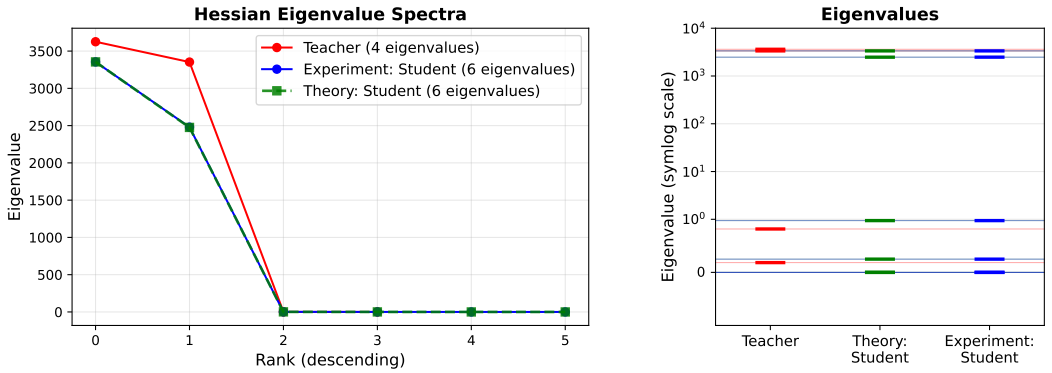


Figure 13: Empirical eigenvalue transformation for a width-2 teacher and width-3 student after optimization. *Left:* ranked eigenvalue spectra. *Right:* eigenvalue comparison. Eigenvalues found experimentally align to within 0.2% of those predicted by Theorem 4.3. There are two additional flat directions from splitting.

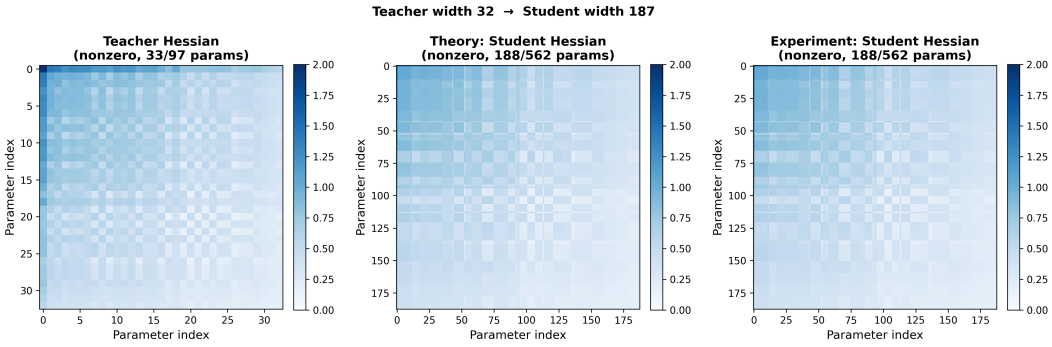


Figure 14: Theoretical and empirical Hessian comparison at larger scale computed by direct application of a symmetry transformation (teacher width 32 $\rightarrow$ student width 187, sigmoid activation). *Left:* teacher Hessian. *Center:* student Hessian predicted by Theorem 4.3. *Right:* empirically measured student Hessian. Hessian entries agree nearly exactly, implying the prediction scales to larger networks.

Character-level Transformer on Alice in Wonderland. The teacher is a 4-layer, 4-head decoder-only Transformer with d_{model}=32, learned absolute positional embeddings, and a 73-character vocabulary, pretrained on the Alice corpus tokenized into 128-character windows. Students share the architecture but vary $d_{\text{model}} \in \{32, 64, 128\}$ and minimize per-token KL divergence to the teacher's logits. We use Adam at 10^{-3}, batch size 256, up to 50,000 epochs, and compute the top 50 Hessian eigenvalues (rather than 100) per student.

M.3 Compute Resources

All experiments are run on NVIDIA L40S GPUs. A single seed of a single configuration takes between roughly 5 minutes (smallest synthetic MLP runs) and 12 hours (largest Transformer student-teacher runs).

M.4 Expansion Checks with Standard Activations

The zero-loss experiments in Section 7.1 use the activation of Simsek et al. (2021) so that every global minimum of the student lies on the teacher's orbit. To confirm that the Hessian relation itself does not depend on this choice, we repeat the expansion with sigmoid and ReLU teachers: we train a teacher, apply a symmetry expansion directly, and compare the Hessian of the expanded network to the prediction of Theorem 4.3. For ReLU the teacher Hessian is singular because of the rescaling

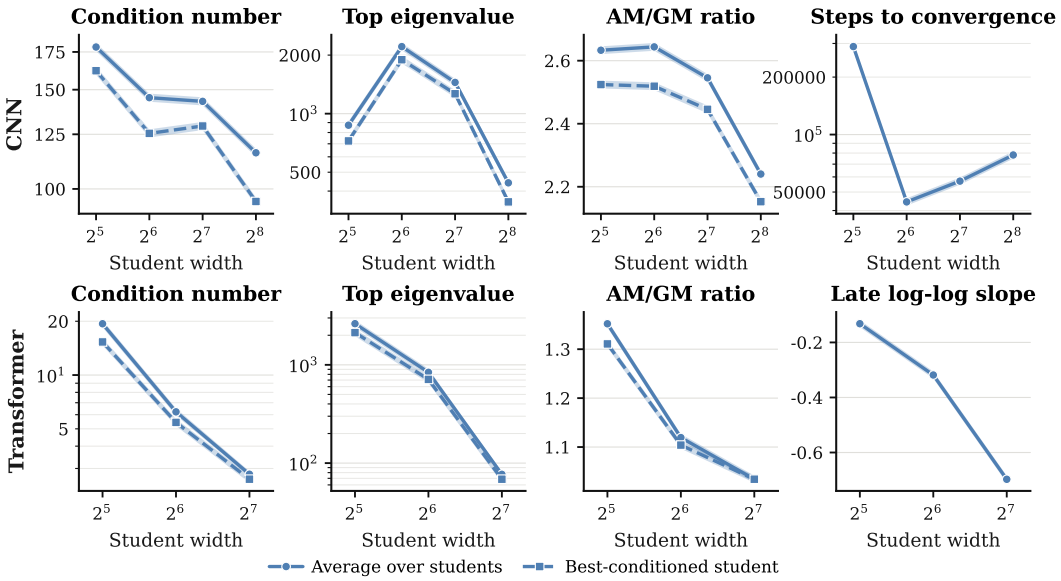


Figure 15: *Top:* CNN student-teacher experiment. A ConvNet teacher (width 32, 4 conv layers) is pretrained on CIFAR-10. Student ConvNets of increasing channel width are trained to match the teacher's output logits. *Bottom:* Transformer student-teacher experiment. A character-level decoder-only Transformer teacher ($d_{\text{model}} = 32$, depth 4) is pretrained on Alice in Wonderland. Student Transformers of increasing width are trained to match the teacher's output distribution via KL divergence. In both settings, condition number, top eigenvalue, and AM/GM ratio improve with width, alongside convergence behavior (steps to convergence for the CNN, late log-log slope for the Transformer).

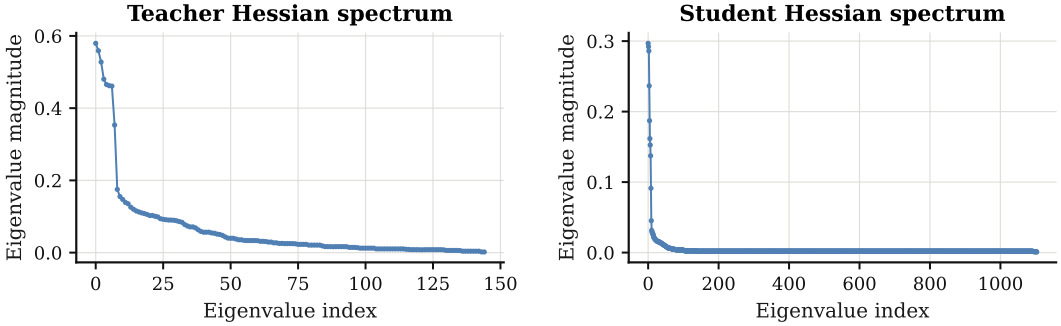


Figure 16: Hessian eigenvalue spectrum for a teacher network (top, width $n_f = 8$) and an overparameterized student (bottom, width $n = 64$). The teacher spectrum decays smoothly, while the student spectrum concentrates its curvature in a small number of leading eigenvalues with the vast majority near zero, corresponding to the flat directions introduced by overparameterization symmetries.

symmetry (rank 9 of 13 for the network used), so we compare the pseudo-determinant against the range-restricted identity of Theorem H.3. Table 1 reports the results; the last column takes the maximum over 24 random uneven splits.

Activation	Base Hessian H	Rel. error, eigenvalues	Rel. error, $\mathrm{pdet}(H')$	Max rel. error, uneven splits
sigmoid	full rank	6×10^{-11}	2×10^{-11}	6×10^{-11}
ReLU	singular, rank 9 of 13	3×10^{-13}	1×10^{-14}	3×10^{-13}

Table 1: Expansion checks with standard activations. Eigenvalues follow Theorem 4.3 in both cases. For sigmoid, pdet follows Theorem 4.4; for ReLU it follows the singular form in Theorem H.3.

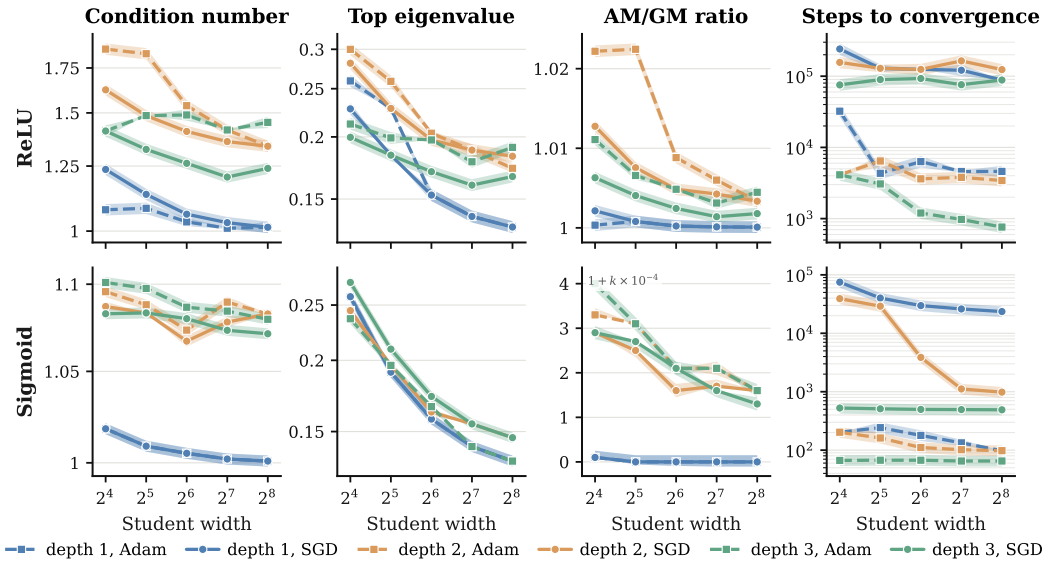


Figure 17: Effect of depth on MLP student-teacher conditioning (teacher width 16). Students at depths 1, 2, and 3 are trained with Adam and SGD (ReLU top, sigmoid bottom), with student width up to 256. Conditioning improves with width at all depths with no clear depth dependence, suggesting the symmetry-based mechanism is robust to depth.

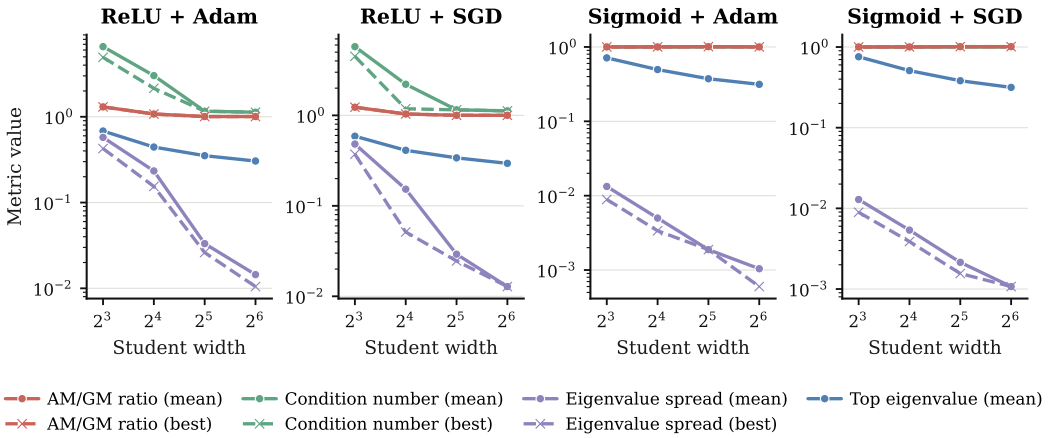


Figure 18: A comparison of conditioning across the average and best-conditioned students. Training is done with 1 teacher and 50 student models across 5 seeds (an extended sweep, distinct from the main MLP setup in Table 5).

We also trained wide sigmoid students from random initialization on a sigmoid teacher. Some runs reach zero loss at a symmetry expansion of the teacher, with the clustering structure of Figure 3; others reach zero loss at global minima that are not on the teacher's orbit, which sigmoid permits. So ordinary training does find the orbit, but with a standard activation the orbit is not the whole minimum set.

M.5 Size of the Gauss-Newton Correction

Away from zero loss the Hessian is $H_{\text{full}} = H_{\text{GN}} + R$, where H_{GN} is the Gauss-Newton term to which the relation $H' = B^\top H B$ applies exactly and R is a correction proportional to the residual that vanishes at zero loss. We measure $\|R\|_F / \|H_{\text{full}}\|_F$ and the relative gap between the condition numbers of H_{full} and H_{GN}, computed on the same top eigenvalues, at several points during training. Table 2 shows a ReLU student-teacher MLP (sigmoid is similar) and Table 3 the CIFAR-10 ConvNet

from Section 7.3. In both cases the correction shrinks as the loss falls, and the effect on conditioning is well under one percent.

Training loss	$\|R\|_F/\|H_{\text{full}}\|_F$	Gap in κ, H_{full} vs. H_{GN}
10^{-2}	4.7%	0.6%
10^{-3}	0.7%	0.05%
10^{-4}	0.16%	0.009%

Table 2: Gauss-Newton correction for a ReLU student-teacher MLP.

Training loss	$\|R\|_F/\|H_{\text{full}}\|_F$	Gap in κ, H_{full} vs. H_{GN}
0.29	9.1%	0.10%
0.10	7.5%	0.21%
0.05	3.8%	0.10%
0.03	1.1%	0.10%

Table 3: Gauss-Newton correction for the CIFAR-10 ConvNet.

M.6 Calibrating the Volume Bound

The bound in Equation (6) contains a constant C that bounds the fourth-order coefficient of the loss along the flat direction opened by a split. We measure it directly on student-teacher MLPs: starting from a global minimum, we move along the split direction, record the loss, and fit a power law. The fitted log-log slope lies between 3.99 and 4.01 across activations, widths, and split fractions, which confirms the $\epsilon^{1/4}$ form of the bound. We take C to be the largest fitted coefficient over split fractions, neurons, and random teachers. Table 4 lists, for each measured C and for some deliberately pessimistic values, the smallest teacher width n_f at which the right-hand side of Equation (40) exceeds one. For the measured constants this happens at a few to a few tens of neurons.

C	$\epsilon = 10^{-2}$	10^{-4}	10^{-6}	10^{-8}	10^{-10}	10^{-12}
1.3×10^{-3} (sigmoid, measured)	1	2	4	6	9	15
1.0×10^{-1} (tanh, measured)	2	4	6	9	15	23
3.9×10^{-1} (softplus + sigmoid, measured)	3	4	7	11	17	27
1 (pessimistic)	3	5	7	12	19	29
10^2 (pessimistic)	5	7	12	19	29	47
10^4 (pessimistic)	7	12	19	29	47	74

Table 4: Smallest teacher width n_f at which the lower bound in Equation (40) exceeds one, for measured and pessimistic values of C.

M.7 Hessian Computation

After convergence, we compute the Hessian eigenspectrum using the Lanczos algorithm with full reorthogonalization, which efficiently approximates the eigenvalue distribution via Hessian-vector products without forming the full matrix. For the small networks used in theory validation, we additionally compute the full Hessian exactly via finite differences to verify the Lanczos estimates. We extract: the complete eigenvalue spectrum, trace, top eigenvalue, condition number ($\lambda_{\max}/\lambda^+_{\min}$), pseudo-determinant (product of nonzero eigenvalues), and the AM/GM ratio of positive eigenvalues. To handle Lanczos numerical noise, eigenvalues below a small magnitude threshold are set to zero. Gradient norms are also recorded at the final iterate.

Parameter	MLP s–t	MLP Housing	CNN*	LM s–t
Architecture	MLP	MLP	ConvNet (4 conv)	CharTransformer
Dataset	synthetic	Cal. Housing	CIFAR-10	Alice in Wonderland
Teacher	random init	none (direct)	pretrained	pretrained
Teacher width n_f	$\{8, 16, 32, 64\}^\dagger$	16	32 channels	32 (d_{model})
Student widths n	$n_f \cdot \{1, 2, 4, 8, 16\}$	$\{16, 32, 64, 128\}$	$\{32, 64, 128, 256\}$	$\{32, 64, 128\}$
Depth	$\{1, 2, 3\}^\dagger$	2	4	4
Input / output dim.	n_f / n_f	8 / 1	3×32^2 / 10	vocab=73
Activation	ReLU, sigmoid	ReLU	ReLU	ReLU FFN, 4 heads
# students	5	25	5	5
Optimizer(s)	SGD, Adam	SGD, Adam	Adam	Adam
LR (SGD / Adam)	0.1 / 0.01	0.1 / 0.01	10^{-3}	10^{-3}
Max epochs (SGD/Adam)	50k / 5k	50k / 5k	5k	50k
Loss	MSE	MSE	MSE / CE*	KL on logits
Batch size	online	200	256	256
Early-stop loss	5×10^{-5}	5×10^{-5}	$0.005/0.4^*$	—
Patience (epochs)	20k	20k	20k	20k
Param.	μP (base 8)	μP (base 8)	standard	standard
Top-k Hessian eigs.	100	100	100	50
Seeds	0, 1, 2	0, 1, 2	0, 1, 2	0, 1, 2

Table 5: Experimental hyperparameters. † For the synthetic MLP student-teacher, depth 1 is swept over $n_f \in \{8, 16, 32, 64\}$ and depths 2–3 over $n_f \in \{8, 16\}$. * CNN sweep covers two settings sharing all hyperparameters except loss/early-stop: *teacher-student* (MSE vs. pretrained ConvNet, stop 0.005) and *direct* (cross-entropy vs. CIFAR-10 labels, stop 0.4). MLP Housing uses 200 standardized real samples; CNN uses the full CIFAR-10 training split; LM uses the Alice corpus tokenized into 128-character non-overlapping windows.

M.8 Hyperparameters

N Notes on Plasticity Loss

A direct application of our framework concerns plasticity loss in continual learning. If optimization converges to a simple generalizing function, the solution lies on a low-functional-width column of the overparameterization groupoid (Figure 7). Such solutions are typically realized through dormant capacity such as duplicated or dead neurons, configurations that are fixed points of subgroups of the network's symmetry group. It is a well-known result that gradient descent preserves these fixed-point subspaces under any loss: cloned neurons stay cloned, dead neurons stay dead, throughout training. Once the objective changes, the optimizer cannot recruit this dormant capacity to fit the new task.

Ziyin (2023) previously analyzed this trapping mechanism for permutation, rescaling, and rotation symmetries under weight decay or noise, and explicitly linked it to plasticity loss. A more detailed and concurrent dynamical systems treatment appears in Joudaki et al. (2025).