# A MATHEMATICAL FRAMEWORK FOR THE HIERARCHICAL ANALYSIS OF NEURAL NETWORKS

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

## ABSTRACT

We introduce a mathematical framework for analyzing and representing neural networks in a way that is both structurally aware and hierarchical. We begin by developing two general-purpose, mathematically-grounded tools for the pairwise analysis of neurons. First, we introduce Structural Component Analysis (SCA) and prove it finds the optimally disentangled orthonormal basis for the joint input-output weight space. First, Structural Component Analysis (SCA) finds a maximally disentangled orthonormal basis for the joint input-output weight space. Second, we formalize the Gated Interaction Decomposition (GID) provides a generative, and generally oblique, basis that explains *how* the coupling between input and output weights arises from an interpretable multiplicative gating mechanism. We then present the Hierarchical Encoding via Reversible Merging (HERM) framework, which leverages GID as a core operator to iteratively and losslessly merge pairs of neurons, thereby transforming a trained network into a compressed skeletal representation and a sequence of recovery instructions. This process reveals a meaningful functional hierarchy of the network's learned knowledge. Our theoretical approach provides a new lens for understanding and manipulating neural networks, and we illustrate its potential through proof-of-concept applications in elastic fine-tuning, dynamic architecture design, and model compression.

## 1 INTRODUCTION

Deep neural networks have achieved remarkable success across a vast range of domains, yet our understanding of their internal workings remains surprisingly limited. As models grow in complexity, they are increasingly treated as opaque "black boxes," making it difficult to interpret their learned representations, debug their failures, or efficiently adapt them to new tasks. A central obstacle here is the lack of analysis tools that can decompose a network's function in a way that respects its inherent structure. For instance, standard linear analysis techniques like Singular Value Decomposition (SVD) applied to a weight matrix ignore the fundamental distinction between a neuron's input weights (which detect features) and its output weights (which propagate signals). Furthermore, most methods provide a "flat" view of the network, failing to uncover the hierarchical relationships that may exist among its learned concepts.

This paper introduces a mathematical framework designed to address these shortcomings by providing a structured, hierarchical, and reversible way to analyze neural networks. Our approach is built from the ground up, starting with the fundamental mathematical unit of a network: the interaction between neurons. We argue that by understanding how pairs of neurons cooperate and specialize, we can build a comprehensive picture of the entire network's function. Our framework consists of three main components. We first introduce two general-purpose tools for the analysis of any pair of neurons within a layer.

**Structural Component Analysis (SCA).** A method to find an orthonormal basis for the subspace spanned by the joint input-output weights of neurons. Critically, this basis is maximally disentangled, meaning its vectors align as closely as possible with the input and output weight spaces, respectively. SCA answers the question: *what is the most structurally pure representation of the function performed by these neurons?*

**Gated Interaction Decomposition (GID).** A more expressive, generative model that provides a specific parameterization for the subspace spanned by two neurons. GID explains *how* the coupling between input and output spaces arises via a multiplicative gating mechanism, modeling a form of second-order conditional logic. It answers the question: *what is the underlying mechanism of interaction between these neurons?*

**Hierarchical Encoding via Reversible Merging (HERM).** A framework that uses GID as a core algebraic operator to iteratively merge pairs of neurons in a provably lossless manner. This process transforms a trained network into a compressed "skeletal" network and a sequence of recovery instructions. This ordered sequence of merges reveals a functional hierarchy, from granular, specialized concepts (merged first) to fundamental, core concepts (merged last).

The main contribution of this paper is not an empirical result but the mathematical framework itself. SCA, GID, and HERM are **general-purpose tools** rooted in sound mathematical foundations, offering a new perspective on network analysis. To demonstrate their potential, we conclude with two minimal, proof-of-concept experiments. We show how the hierarchy revealed by HERM can be used to guide a principled form of elastic fine-tuning for transfer learning and how its reversible operators can provide the mechanics for creating dynamic, self-organizing architectures that grow and shrink during training, and finally how partial decoding can be used for model compresion. These applications underscore the utility of a framework that goes beyond passive analysis to enable active and intelligent manipulation of neural network structure. **All relevant algorithms and mathematical proofs are provided in the supplementary appendix**.

## 2 RELATED WORKS

**Structured Subspace Analysis** Classical subspace analysis methods like PCA Pearson (1901); Hotelling (1933) or SVD Eckart & Young (1936) are structure-agnostic, mixing functionally distinct components. Our **SCA** is instead designed to find an orthonormal basis for the joint space $\mathbb{R}^n \times \mathbb{R}^c$ that is maximally disentangled, respecting this partition. This differs from two-view methods like Canonical Correlation Analysis (CCA) Hotelling (1936) or Partial Least Squares (PLS) Wold (1975). These methods, including modern variants Hardoon et al. (2004); Witten et al. (2009); Andrew et al. (2013), seek to *relate* two *separate* subspaces (e.g., span($\mathbf{W}_{\text{in}}$) and span($\mathbf{W}_{\text{out}}$)), whereas SCA analyzes a *single* joint subspace $S$ and finds a single, maximally-aligned basis for it. This line of inquiry remains active in the analysis of modern representation geometry Hoogeboom et al. (2023); Pimentel et al. (2024).

**Interpretable Bases and Multiplicative Interactions** Our **GID** produces a generally non-orthogonal (oblique) basis. This connects it to methods like Independent Component Analysis (ICA) Comon (1994), Non-negative Matrix Factorization (NMF) Lee & Seung (1999), and sparse coding Olshausen & Field (1997); Aharon et al. (2006). However, while these methods derive obliqueness from statistical priors (e.g., independence, sparsity), GID's obliqueness is a necessary consequence of its specific, mechanistic hypothesis: a multiplicative, second-order gating process. This goal of finding interpretable, non-orthogonal "features" is shared by modern dictionary learning approaches used to analyze LLMs Lanchantin et al. (2023); Adly & et al. (2024). The core GID mechanism, $\mathbf{p}_c = (\langle \mathbf{u}, \mathbf{z} \rangle)\, \mathbf{r}$, is a form of bilinear interaction, a concept related to tensor decompositions Harshman (1970); Kolda & Bader (2009) and ubiquitous in deep learning. Gating mechanisms are foundational to LSTMs Hochreiter & Schmidhuber (1997) and dot-product attention Vaswani et al. (2017), and remain central to state-of-the-art LLMs through innovations like Rotary Position Embeddings (RoPE) Su et al. (2024) and Grouped-Query Attention (GQA) Ainslie et al. (2023). GID contributes a novel post-hoc *analytical tool* to interpret this coupling in static, learned weights.

**Network Interpretability and Neuroscience Parallels** Our framework offers a new approach to network interpretability. It analyzes neither activations Olah et al. (2017); Bau et al. (2017); Kornblith et al. (2019) nor single weight matrices Saxe et al. (2013); Han et al. (2015) in isolation. Instead, SCA and GID are, to our knowledge, the first tools designed specifically to model the *joint input-output weight space*. This allows a decomposition of function based on the coupling between what a neuron detects (input) and how it contributes (output), aligning with the goals of mechanistic interpretability Nanda et al. (2023); Olah et al. (2020). This approach has strong parallels in computational neuroscience. **GID**'s multiplicative gating provides a mathematical form for dendritic computation, where biological neurons perform complex, non-linear operations in their dendrites Mel (1994); London & Häusser (2005); Gidon et al. (2020); Häusser & Magee (2021). **HERM**'s discovery of a functional hierarchy mirrors the principles of efficient and hierarchical coding in sensory pathways Barlow (1961); Olshausen & Field (1996); Felleman & Van Essen (1991), while its merge operation is a computational analog for the formation of neural assemblies Buzsáki (2010); Holtmaat et al. (2022). Finally, **SCA**'s joint analysis of input-output structure resonates with the central goal of modern connectomics: linking structural wiring diagrams to function Sporns (2010); Bullmore & Sporns (2009); Shapson-Coe et al. (2024).

## 3 Structural Component Analysis (SCA)

We introduce Structural Component Analysis (SCA) to analyze data possessing an intrinsic binary partition, specifically the joint input and output weight parameters of neurons in a network. Consider a layer with $h$ neurons, each of which has $n$ inputs, and $c$ outputs. The parameters for each neuron $1 \le i \le h$ are represented by a joint vector $\mathbf{w}_i \in \mathbb{R}^{n+c}$, formed by concatenating its input weights ($\in \mathbb{R}^n$) and output weights ($\in \mathbb{R}^c$). These $h$ vectors span a $k$-dimensional subspace $S \triangleq \text{span}(\{\mathbf{w}_i\}_{i=1}^h) \subseteq \mathbb{R}^{n+c}$, where $k \le h$ is the rank.

To reveal a specific structure within the vectors $\{\mathbf{w}_i\}_{i=1}^h$, one can apply the appropriate orthogonal transform—a rotation of the coordinate system—to find a new basis where each axis has a special meaning. Examples include the Discrete Fourier Transform (DFT) and Principal Component Analysis (PCA), which find spaces well-suited for frequency and variance analysis, respectively. These transforms, however, are structure-agnostic. They ignore an inherent $\mathbb{R}^n \times \mathbb{R}^c$ partition in the data, producing basis vectors that arbitrarily mix these components. This mixing is problematic because the two subspaces are functionally distinct. In a neural network, for example, a neuron's input weights $\in \mathbb{R}^n$ act as a **feature detector** (a template or filter), while its output weights $\in \mathbb{R}^c$ translate the detection result into a **task-relevant signal** to the next layer. A basis that merges these two separate functions is difficult to interpret.

SCA addresses this by finding an orthonormal basis $\{\mathbf{b}_1, \dots, \mathbf{b}_k\}$ for $S$ that is maximally **structurally disentangled**. That is, it seeks basis vectors that are either n-dominated or c-dominated, while penalizing vectors that mix components from both spaces. Formally, given a hyperparameter $k_n$ (denoting the desired number of n-dominated basis vectors, where $1 \le k_n < k$), SCA seeks to find the basis $\{\mathbf{b}_i\}$ that minimizes the total **structural coupling cost** $J$. This cost function quantifies the cumulative "spill-over" of basis vectors into their non-dominant subspace $J(\{\mathbf{b}_i\}_{i=1}^k) = \sum_{i=1}^{k_n} \|(\mathbf{b}_i)_c\|^2 + \sum_{j=k_n+1}^{k} \|(\mathbf{b}_j)_n\|^2$, where $(\mathbf{b})_n \in \mathbb{R}^n$ and $(\mathbf{b})_c \in \mathbb{R}^c$ are the components of $\mathbf{b}$. Minimizing $J$ forces the first $k_n$ vectors to align as closely as possible with the $\mathbb{R}^n$ subspace and the remaining $k - k_n$ vectors to align with the $\mathbb{R}^c$ subspace, while collectively remaining an orthonormal basis for $S$. See Figure 1 for a visual illustration.

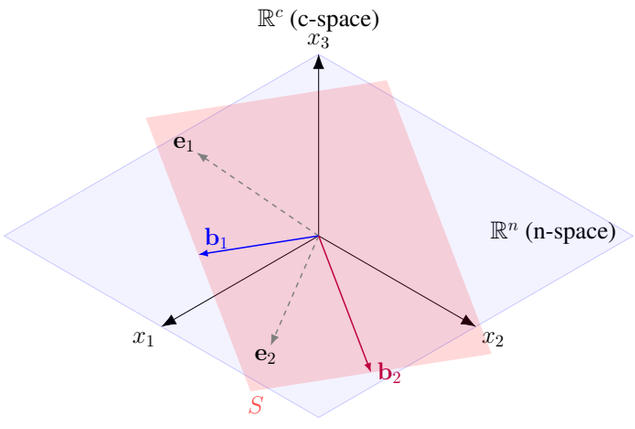


Figure 1: Visualizations of the Structural Component Analysis (SCA). The blue-shaded region is the n-space ($\mathbb{R}^n$, visualized as a 2D plane) and the vertical z-axis represents the c-space ($\mathbb{R}^c$, visualized as 1D). The red-shaded region is the subspace $S$. SCA finds an n-dominated vector $\mathbf{b}_1$ (blue) that lies as flat as possible within $S$, and a c-dominated vector $\mathbf{b}_2$ (purple) that is orthogonal to it and maximally aligned with the c-space. A conventional basis like $\{\mathbf{e}_1, \mathbf{e}_2\}$ is less disentangled.

The optimal basis $\{\mathbf{b}_i^*\}$ that minimizes the structural coupling cost $J$ is guaranteed to exist and can be constructed via the following procedure (see Proposition 2 for proof). We first choose any arbitrary orthonormal basis $\{\mathbf{e}_1, \dots, \mathbf{e}_k\}$ for the $k$-dimensional subspace $S$ (e.g., via SVD on the matrix $\mathbf{W}$ whose rows are $\mathbf{w}_i^\top$). Then we construct the $k \times k$ symmetric, positive semi-definite matrix $\mathbf{M}$ using only the **n-space components** of the initial basis $\mathbf{M}_{ij} = (\mathbf{e}_i)_n^\top (\mathbf{e}_j)_n$. This matrix captures the projection of the subspace $S$ onto the $\mathbb{R}^n$ space, as seen from the basis $\{\mathbf{e}_i\}$. We now compute the complete eigendecomposition of $\mathbf{M}$, yielding $k$ orthonormal eigenvectors $\mathbf{v}_1, \dots, \mathbf{v}_k \in \mathbb{R}^k$ and their corresponding real eigenvalues $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_k \ge 0$, that is $\mathbf{M}\mathbf{v}_j = \lambda_j \mathbf{v}_j \quad \forall j \in \{1, \dots, k\}$. At the end we construct the SCA basis vectors $\{\mathbf{b}_1^*, \dots, \mathbf{b}_k^*\}$ by linear combinations of the initial basis vectors, using the components of the eigenvectors of $\mathbf{M}$ as coefficients. That is, the new basis is a rotation of the old basis, defined by the orthogonal matrix $\mathbf{V} = [\mathbf{v}_1 | \dots | \mathbf{v}_k]$, which yields $\boldsymbol{b}_j^* = \sum_{i=1}^k (\mathbf{v}_j)_i \, \mathbf{c}_i$, where $(\mathbf{v}_j)_i$ is the $i$-th component of $\mathbf{v}_j$. The first $k_n$ vectors,

$\{\mathbf{b}_1^*, \ldots, \mathbf{b}_{k_n}^*\}$, corresponding to the $k_n$ largest eigenvalues, constitute the optimal set of n-dominated vectors. The remaining vectors, $\{\mathbf{b}_{k_n+1}^*, \ldots, \mathbf{b}_k^*\}$, are the optimal c-dominated vectors. The eigenvalues themselves quantify the disentanglement: $\lambda_j = \left\|(\mathbf{b}_j^*)_n\right\|^2$. Since $\left\|\mathbf{b}_j^*\right\|^2 = 1$ (as it's an orthonormal basis), this implies $\left\|(\mathbf{b}_j^*)_c\right\|^2 = 1 - \lambda_j$. Thus, eigenvalues close to 1 correspond to basis vectors residing almost entirely in $\mathbb{R}^n$, while eigenvalues close to 0 correspond to vectors almost entirely in $\mathbb{R}^c$. This allows us to categorize the nature of the coupling as: **No Interaction:** All eigenvalues $\lambda_j$ are either 1 or 0. The subspace $S$ fully decomposes into a direct sum of a subspace within $\mathbb{R}^n$ and a subspace within $\mathbb{R}^c$. **One-Sided Interaction:** One set of coupling terms is zero, but the other is not. **Two-Sided Interaction:** The general case, where both n-dominated and c-dominated vectors have non-zero "spill-over" (i.e., eigenvalues $0 < \lambda_j < 1$ exist).

### 3.1 The Two-Dimensional Case: $\dim(S) = 2$

**Motivation** Analyzing the two-dimensional case for a subspace $S \subseteq \mathbb{R}^{n+c}$ is crucial. While focusing on a 2D subspace may seem restrictive, this approach allows us to analyze an entire deep network by iteratively applying SCA (or its derivative, GID) to pairs of neurons. The computational implications of this pairwise method are profound. It reduces the most expensive operations—eigendecomposition and, for GID, matrix inversion—from large $(n+c)\times(n+c)$ matrices to trivial $2 \times 2$ matrices. This is the **miracle**: for the $2 \times 2$ case, both operations have simple, closed-form algebraic solutions. Consequently, we can completely avoid the iterative and computationally costly numerical procedures required for general-sized matrices.

**Notation** When $\dim(S) = 2$, the only meaningful choice for the SCA parameters is $k = 2$ and $k_n = 1$. For clarity, we denote the two resulting SCA basis vectors as $\mathbf{p}^*$ and $\mathbf{q}^*$ instead of $\boldsymbol{b}_1^*$ and $\boldsymbol{b}_2^*$. Without loss of generality, we let $\mathbf{p}^*$ be $n$-dominated and $\mathbf{q}^*$ be $c$-dominated. Accordingly, the subvectors $\mathbf{p}_n^*$ and $\mathbf{q}_c^*$ are called the *primary* components, while the subvectors $\mathbf{p}_c^*$ and $\mathbf{q}_n^*$ are the *coupling* components, which represent the structural coupling.

## 4 Gated Interaction Decomposition (GID)

### 4.1 Core Definition and Motivation

The Gated Interaction Decomposition (GID) is a specialized parameterization for any $S \subseteq \mathbb{R}^{n+c}$ with $\dim(S) = 2$. Recall that SCA finds an optimally disentangled basis. While SCA's objective function $J$ is bounded below by zero, the actual minimum value $J^*$ is typically non-zero, meaning the resulting basis still retains an inevitable and irreducible residual coupling. This coupling is crucial for understanding the underlying process that generated the subspace, yet SCA only provides the basis; it does not explain the mechanism of this coupling. GID, in contrast, constructs a specific, and generally non-orthogonal, basis $\{\mathbf{p}, \mathbf{q}\}$ from four underlying primitive vectors $\{\mathbf{u}, \mathbf{v}, \mathbf{z}, \mathbf{r}\}$ where $\|\mathbf{u}\| = \|\mathbf{v}\| = 1$. We refer to $\mathbf{u} \in \mathbb{R}^n$ and $\mathbf{v} \in \mathbb{R}^c$ as **primary vectors** and to $\mathbf{z} \in \mathbb{R}^n$ and $\mathbf{r} \in \mathbb{R}^c$ as **interaction channels**. These primitives form the basis vectors according to a precise multiplicative structure:

$$\mathbf{p} = \begin{bmatrix} \mathbf{u} \\ (\langle \mathbf{u}, \mathbf{z} \rangle)\, \mathbf{r} \end{bmatrix} \quad \text{and} \quad \mathbf{q} = \begin{bmatrix} (\langle \mathbf{v}, \mathbf{r} \rangle)\, \mathbf{z} \\ \mathbf{v} \end{bmatrix} \tag{1}$$

It is precisely this structure that characterizes how the spaces interact—i.e., how the n-space influences the c-space, and vice versa. The scalar terms $\langle \mathbf{u}, \mathbf{z} \rangle$ and $\langle \mathbf{v}, \mathbf{r} \rangle$ act as **gates** that control the magnitude of this cross-space coupling. Specifically, the interaction mechanism is as follows: In the basis vector $\mathbf{p}$, the c-space component $\mathbf{r}$ is gated by the interaction between the n-space primary vector $\mathbf{u}$ and the n-space channel $\mathbf{z}$. In the basis vector $\mathbf{q}$, the n-space component $\mathbf{z}$ is gated by the interaction between the c-space primary vector $\mathbf{v}$ and the c-space channel $\mathbf{r}$.

### 4.2 GID Interaction Modes

The same taxonomy of interaction modes developed for SCA (no interaction, one-sided, two-sided) can also be applied to GID. In fact, the modes derived from both methods must be identical. This is because the interaction mode is a fundamental property of the subspace S itself, not of the particular basis chosen to represent it (see Propositions 3 and 4). Within this taxonomy, two-sided interactions represent the most general and complex form, making them the most practically relevant mode. However, one-sided interaction spaces, when represented by GID, exhibit special properties that make them particularly attractive from a theoretical perspective. Given the distinct practical and theoretical importance of each, our analysis will cover both modes.

### 4.3 One-Sided Interactions: Axiomatic Foundations and Simple Solution Forms

While the GID basis $\{\mathbf{p}, \mathbf{q}\}$ is generally non-orthogonal (and thus distinct from the orthonormal SCA basis $\{\mathbf{p}^*, \mathbf{q}^*\}$), in the special case of one-sided interaction spaces, the two bases are proven to be parallel. This connection is highly useful. It stems from the definition of a one-sided SCA basis, in which one coupling component is zero (e.g., $(\mathbf{q}^*)_n = \mathbf{0}$).This perfect alignment allows for the derivation of a **simple analytical solution** for a set of GID primitives, computed directly from the SCA basis components. One-sided interaction spaces are of special interest from theoretical perspective for the following reasons:

**Orthogonality and Simple Solution Form for Primitives** . While the GID basis $\{\mathbf{p}, \mathbf{q}\}$ is generally non-orthogonal (and thus distinct from the orthonormal SCA basis $\{\mathbf{p}^*, \mathbf{q}^*\}$), in the special case of one-sided interaction spaces, the two bases are proven to be parallel. This connection is highly useful. It stems from the definition of a one-sided SCA basis, in which one coupling component is zero (e.g., $(\mathbf{q}^*)_n = \mathbf{0}$).This perfect alignment allows for the derivation of a **simple analytical solution** for a set of GID primitives, computed directly from the SCA basis components. For the case where $(\mathbf{q}^*)_n = \mathbf{0}$: $\mathbf{u} \triangleq \frac{\mathbf{p}_n^*}{\|\mathbf{p}_n^*\|}$ , $\mathbf{v} \triangleq \mathbf{q}_c^*$ , $\mathbf{z} \triangleq \frac{\mathbf{p}_n^*}{\|\mathbf{p}_n^*\|^2}$ , $\mathbf{r} \triangleq \mathbf{p}_c^*$. This solution has a continuous scaling ambiguity, as the set $\{s\mathbf{z}, (1/s)\mathbf{r}\}$ for any $s \neq 0$ describes the same subspace. See proof of Theorem 5 for details.

**Axiomatic Foundations: Why GID is special to represent a 2D subspace over other possibilities?** The GID structure is not an arbitrary choice but is the **unique** mathematical representation that satisfies three intuitive principles for a 2D basis $\{\mathbf{p}, \mathbf{q}\}$:

(1) *Minimal Structural Coupling.* The basis must be maximally disentangled, as found by SCA.

(2) *Minimal Interaction Rank.* Let the coupling between spaces be defined as linear maps $L_{n\to c} : \mathbb{R}^n \to \mathbb{R}^c$ and $L_{c\to n} : \mathbb{R}^c \to \mathbb{R}^n$, where $\mathbf{p}_c = L_{n\to c}(\mathbf{p}_n)$ and $\mathbf{q}_n = L_{c\to n}(\mathbf{q}_c)$. This principle mandates minimal interaction by requiring $L_{n\to c}$ and $L_{c\to n}$ to be the simplest possible non-trivial form: a **rank-1 operator**.

(3) *Shared Interaction Mechanism.* The coupling is symmetric, sharing the same interaction channels $\{\mathbf{z}, \mathbf{r}\}$.

**Any basis satisfying these three axioms must take the GID parametric form.** See Figure 2 for illustration and Theorem 6 for details.

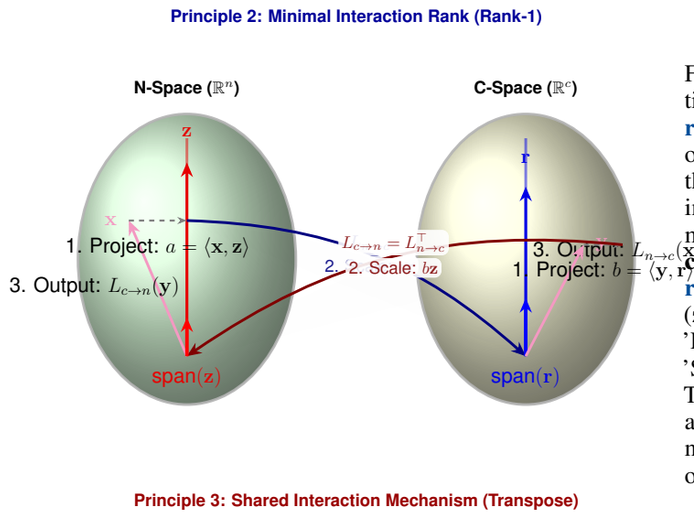


Figure 2: A visualization of the GID interaction axioms. **[Principle 2] The map $L_{n\to c}$ is a rank-1 bottleneck**: An input $\mathbf{x}$ is (1) 'Projected' onto the $\mathbf{z}$ channel to get a scalar $a$, which is then (2) used to 'Scale' the $\mathbf{r}$ vector, producing (3) the final 'Output' vector. The output must lie along the 1D line defined by $\mathbf{r}$. **[Principle 3] The map $L_{c\to n}$ is a shared, symmetric mechanism**: It uses the *exact same vectors* $(\mathbf{z}, \mathbf{r})$ but in reversed roles. An input $\mathbf{y}$ is (1) 'Projected' onto $\mathbf{r}$ to get $b$, which is (2) used to 'Scale' $\mathbf{z}$, producing (3) the final 'Output' vector. This operational symmetry, where the projection and scaling channels are swapped, is the visual meaning of the maps being 'Transposes' of each other ($L_{c\to n} = L_{n\to c}^\top$).

### 4.4 The Two-Sided Interaction Case: Solution via Quartic Root-Finding

While two-sided interactions are more general and typical in real-world applications, this generalization introduces significant analytical challenges. The jump from one-sided to two-sided interaction spaces is not trivial; we lose the clear axiomatic foundation that made GID the unique and necessary representation for the one-sided case. While not a formal proof, the fact that GID arises from first principles in the one-sided case provides a strong justification for its continued use as a general and principled representation. Furthermore, while a closed-form expression for the GID primitives is still attainable, its derivation is substantially more complex. The reason is that a GID basis for a two-sided space is **necessarily oblique** as shown in Proposition 9. This breaks the link to the SCA basis that are by definition **orthonormal**. The remainder of this section is dedicated to developing the mathematical results required to derive these analytical expressions for the two-sided interaction space.

**Roadmap.** First, we formulate the problem as a constrained optimization guided by the principle of minimum energy. Second, we prove that this principle confines the search space for the interaction channels to a low-dimensional subspace, allowing us to project the entire problem into $\mathbb{R}^2$. Third, we derive the *complete* system of constraints that govern the 2D problem, a crucial step whose omission leads to an incorrect, underdetermined solution. Fourth, we show how this fully-constrained system can be analytically collapsed into a univariate optimization problem whose solution is found by finding the roots of a quartic polynomial. Finally, we detail the full algorithm, including critical numerical stability enhancements.

#### 4.4.1 Problem Formulation

Given a pair of neurons with joint weight vectors $\{\mathbf{w}_i, \mathbf{w}_j\}$, we form the joint weight matrix $\mathbf{W} = [\mathbf{w}_i | \mathbf{w}_j] \in \mathbb{R}^{(n+c)\times 2}$. We say a neuron pair is **admissible** if their corresponding input and output weight matrices, $\mathbf{W}_{in} = [(\mathbf{w}_i)_n | (\mathbf{w}_j)_n]$ and $\mathbf{W}_{out} = [(\mathbf{w}_i)_c | (\mathbf{w}_j)_c]$, are both of full column rank (i.e., $\operatorname{rank}(\mathbf{W}_{in}) = \operatorname{rank}(\mathbf{W}_{out}) = 2$). The analysis throughout Section 4.4 assumes admissibility. As demonstrated in Proposition 15, any pair that fails this condition degenerates to a one-sided subspace, which is fully addressed by the methods developed in Section 4.3.

The GID factorization seeks primitives $\mathbf{u}, \mathbf{v}, \mathbf{z}, \mathbf{r}$ and coefficients $\boldsymbol{\alpha}, \boldsymbol{\beta}$ that provide a **perfect reconstruction**. Among the potentially infinite set of solutions, we seek the one whose interaction channels induce **minimum energy**. Formally, we solve:

$$\min_{\{\mathbf{u},\mathbf{v},\mathbf{z},\mathbf{r},\boldsymbol{\alpha},\boldsymbol{\beta}\}} \left(\|\mathbf{z}\|_2^2 + \|\mathbf{r}\|_2^2\right) \quad \text{s.t.} \quad \mathbf{W} = \underbrace{\begin{bmatrix} \mathbf{u} \\ \langle \mathbf{u}, \mathbf{z} \rangle \, \mathbf{r} \end{bmatrix}}_{\mathbf{p}} \boldsymbol{\alpha}^\top + \underbrace{\begin{bmatrix} \langle \mathbf{v}, \mathbf{r} \rangle \, \mathbf{z} \\ \mathbf{v} \end{bmatrix}}_{\mathbf{q}} \boldsymbol{\beta}^\top \text{ and } \|\mathbf{u}\| = \|\mathbf{v}\| = 1 \tag{2}$$

#### 4.4.2 Dimensionality Reduction

The key to an analytical solution is to prove that the search for the optimal interaction channels is not over the entirety of $\mathbb{R}^n$ and $\mathbb{R}^c$, but is restricted to the 2D subspaces defined by the neuron weights themselves. We start by showing the optimal solution to 2 must have its interaction channels $\mathbf{z}$ and $\mathbf{r}$ for in the column space of $\operatorname{span}(\mathbf{W}_{in})$ and $\operatorname{span}(\mathbf{W}_{in})$ respectively per Proposition 10. Consequently, we can parameterize the high-dimensional vectors using a low-dimensional coordinate system. Let the (thin) QR factorizations of the weight matrices be $\mathbf{W}_{in} = \mathbf{Q}_n \mathbf{R}_n$ and $\mathbf{W}_{out} = \mathbf{Q}_c \mathbf{R}_c$, where $\mathbf{Q}_n \in \mathbb{R}^{n\times 2}$ and $\mathbf{Q}_c \in \mathbb{R}^{c\times 2}$ are orthonormal bases for the respective 2D subspaces. We can express the primitives as $\mathbf{z} = \mathbf{Q}_n \mathbf{z}_{2d}$, $\mathbf{r} = \mathbf{Q}_c \mathbf{r}_{2d}$, $\mathbf{u} = \mathbf{Q}_n \mathbf{u}_{2d}$, $\mathbf{v} = \mathbf{Q}_c \mathbf{v}_{2d}$, where the subscript $2d$ means the vector is 2-dimensional. Substituting these into the GID equations and left-multiplying by $\mathbf{Q}_n^\top$ and $\mathbf{Q}_c^\top$ transforms the high-dimensional problem into an equivalent one in $\mathbb{R}^2$:

$$\mathbf{R}_n = \mathbf{u}_{2d} \boldsymbol{\alpha}^\top + \mathbf{z}_{2d} (\mathbf{r}_{2d}^\top \mathbf{v}_{2d}) \boldsymbol{\beta}^\top \tag{3}$$

$$\mathbf{R}_c = \mathbf{v}_{2d} \boldsymbol{\beta}^\top + \mathbf{r}_{2d} (\mathbf{u}_{2d}^\top \mathbf{z}_{2d}) \boldsymbol{\alpha}^\top \tag{4}$$

#### 4.4.3 The Complete System of Constraints in the 2D Subspace

Within this 2D space, the problem becomes tractable. We first define the coefficient matrix $\mathbf{C} \triangleq [\boldsymbol{\alpha} | \boldsymbol{\beta}] \in \mathbb{R}^{2\times 2}$, which Proposition 11 proves is invertible. This invertibility is key, as it allows us to solve for $\mathbf{C}^\top$ and equate the two expressions in (3) and (4). Then simple algebraic manipulation reveals that the GID primitives in the 2D projected space must satisfy two core constraints: $\mathbf{z}_{2d}^\top \mathbf{P} \mathbf{r}_{2d} = 1$ and $\mathbf{z}_{2d}^\top (\mathbf{P}^{-1})^\top \mathbf{r}_{2d} = 1$, where $\mathbf{P} \triangleq \mathbf{R}_n \mathbf{R}_c^{-1}$. See Proposition 12

for detailed derivation steps. Note that the $2 \times 2$ matrix $\mathbf{P} \triangleq \mathbf{R}_n \mathbf{R}_c^{-1}$ is well-defined and invertible. That is because the matrices $\mathbf{R}_n$ and $\mathbf{R}_c$ are upper triangular and full rank by construction from the QR decomposition of linearly independent neuron weights.

#### 4.4.4 Conversion to Unconstrained Optimization

The optimization problem is now reduced to solving the following in $\mathbb{R}^2$: $\min_{\mathbf{z}_{2d}, \mathbf{r}_{2d}} E$ s.t. $\mathbf{z}_{2d}^\top \mathbf{P} \mathbf{r}_{2d} = 1 \quad , \quad \mathbf{z}_{2d}^\top (\mathbf{P}^{-1})^\top \mathbf{r}_{2d} = 1$, where $E \triangleq \|\mathbf{z}_{2d}\|^2 + \|\mathbf{r}_{2d}\|^2$. The constraints form a linear system for $\mathbf{z}_{2d}$, which allows us to solve for it in terms of $\mathbf{r}_{2d}$. Substituting this solution back into the energy function reduces the problem to the following unconstrained optimization over the single 2D vector variable $\mathbf{r}_{2d}$. This is proved in Proposition 13.

$$\min_{\mathbf{r}_{2d} \in \mathbb{R}^2} E(\mathbf{r}_{2d}) = \frac{\mathbf{r}_{2d}^\top (\mathbf{A}^\top \mathbf{A}) \mathbf{r}_{2d}}{(\mathbf{r}_{2d}^\top \mathbf{M} \mathbf{r}_{2d})^2} + \mathbf{r}_{2d}^\top \mathbf{r}_{2d} \,, \tag{5}$$

where $\mathbf{A} = \mathbf{P} - (\mathbf{P}^{-1})^\top$ and $\mathbf{M} = \mathbf{A}^\top \mathbf{J}^\top \mathbf{P}$.

#### 4.4.5 Optimal Length

We now show that the optimal magnitude of the vector $\mathbf{r}_{2d}$ can be determined analytically for any given direction. Let, $\mathbf{r}_{2d} = \rho \cdot \hat{\mathbf{u}} \quad , \quad \|\hat{\mathbf{u}}\| = 1$. By substituting $\mathbf{r}_{2d} = \rho \hat{\mathbf{u}}$ into Eq. (5), the energy becomes $E(\rho) = \frac{\rho^2 (\hat{\mathbf{u}}^\top \mathbf{A}^\top \mathbf{A} \hat{\mathbf{u}})}{\rho^4 (\hat{\mathbf{u}}^\top \mathbf{M} \hat{\mathbf{u}})^2} + \rho^2$. The quadratic forms in $\hat{\mathbf{u}}$ are constant for a fixed direction. Taking the derivative $\frac{dE}{d\rho}$ and setting it to zero yields $-2\rho^{-3} \frac{N}{D^2} + 2\rho = 0$, where $N$ and $D$ are the constant quadratic forms. Solving for $\rho$ gives the optimal magnitude:

$$\rho^4 = \frac{\hat{\mathbf{u}}^\top (\mathbf{A}^\top \mathbf{A}) \hat{\mathbf{u}}}{(\hat{\mathbf{u}}^\top \mathbf{M} \hat{\mathbf{u}})^2} \tag{6}$$

#### 4.4.6 Collapsed 1D Objective Function

In this section, we show that by substituting this optimal magnitude back into the energy function, the problem collapses from a bivariate optimization over $\mathbf{r}_{2d}$ to a simpler univariate optimization over its direction, represented by an angle $\theta$. To obtain the collapsed 1D objective function, we start by showing that the minimum $E$, as a function of the direction vector $\hat{\mathbf{u}} \in \mathbb{S}^1$, is given by:

$$E(\hat{\mathbf{u}}) = 2 \cdot \frac{\sqrt{\hat{\mathbf{u}}^\top (\mathbf{A}^\top \mathbf{A}) \hat{\mathbf{u}}}}{|\hat{\mathbf{u}}^\top \mathbf{M} \hat{\mathbf{u}}|} \tag{7}$$

To see that, let $N(\hat{\mathbf{u}}) = \hat{\mathbf{u}}^\top (\mathbf{A}^\top \mathbf{A}) \hat{\mathbf{u}}$ and $D(\hat{\mathbf{u}}) = \hat{\mathbf{u}}^\top \mathbf{M} \hat{\mathbf{u}}$, so that (6) can be expressed more compactly as $\rho^2 = \frac{\sqrt{N(\hat{\mathbf{u}})}}{|D(\hat{\mathbf{u}})|}$. Substitute this into the energy function $E = \rho^{-2} \frac{N(\hat{\mathbf{u}})}{D(\hat{\mathbf{u}})^2} + \rho^2$. The first term becomes $\frac{|D(\hat{\mathbf{u}})|}{\sqrt{N(\hat{\mathbf{u}})}} \frac{N(\hat{\mathbf{u}})}{D(\hat{\mathbf{u}})^2} = \frac{\sqrt{N(\hat{\mathbf{u}})}}{|D(\hat{\mathbf{u}})|}$. The second term is $\rho^2 = \frac{\sqrt{N(\hat{\mathbf{u}})}}{|D(\hat{\mathbf{u}})|}$. Summing these two identical terms gives the final expression. The problem is now reduced to finding the unit vector $\hat{\mathbf{u}} \in \mathbb{S}^1$ that minimizes Eq. (7). This is equivalent to minimizing its square: $\min_{\hat{\mathbf{u}} \in \mathbb{R}^2, \|\hat{\mathbf{u}}\|=1} F(\hat{\mathbf{u}}) = \frac{\hat{\mathbf{u}}^\top (\mathbf{A}^\top \mathbf{A}) \hat{\mathbf{u}}}{(\hat{\mathbf{u}}^\top \mathbf{M} \hat{\mathbf{u}})^2}$. To solve this, we parameterize the unit vector by an angle, $\hat{\mathbf{u}}(\theta) = [\cos\theta, \sin\theta]^\top$. Let $\mathbf{C} = \mathbf{A}^\top \mathbf{A}$ (a symmetric matrix) and $\mathbf{M}_{sym} = \frac{1}{2}(\mathbf{M} + \mathbf{M}^\top)$, noting that $\hat{\mathbf{u}}^\top \mathbf{M} \hat{\mathbf{u}} = \hat{\mathbf{u}}^\top \mathbf{M}_{sym} \hat{\mathbf{u}}$. A general quadratic form $\hat{\mathbf{u}}^\top \mathbf{X} \hat{\mathbf{u}}$ for a $2 \times 2$ symmetric matrix $\mathbf{X}$ and unit vector $\hat{\mathbf{u}}$ can be written as: $\hat{\mathbf{u}}^\top \mathbf{X} \hat{\mathbf{u}} = \frac{X_{11}+X_{22}}{2} + \frac{X_{11}-X_{22}}{2} \cos(2\theta) + X_{12} \sin(2\theta)$. We apply this to our numerator and denominator $N(\theta) = \hat{\mathbf{u}}^\top \mathbf{C} \hat{\mathbf{u}} = k_{c1} + k_{c2} \cos(2\theta) + k_{c3} \sin(2\theta)$, $D(\theta) = \hat{\mathbf{u}}^\top \mathbf{M}_{sym} \hat{\mathbf{u}} = k_{m1} + k_{m2} \cos(2\theta) + k_{m3} \sin(2\theta)$ and therefore obtain $F(\hat{\mathbf{u}}(\theta)) = \frac{N(\theta)}{D(\theta)}$ where the coefficients $k_{ij}$ are derived directly from the elements of $\mathbf{C}$ and $\mathbf{M}_{sym}$.

#### 4.4.7 Conversion to a Polynomial

To find a purely analytical solution, we reframe the problem from minimizing a trigonometric objective function in $\theta$ to finding the roots of a degree-4 univariate polynomial, which can be solved using known closed-form expressions. To do this, we continue with these following steps.

**Step 1: Set the Derivative to Zero.** To minimize $F(\theta) = N(\theta)/D(\theta)^2$, we set its derivative to zero. Using the quotient rule, $\frac{dF}{d\theta} = 0$ implies that the numerator of the derivative must be zero: $N'(\theta)D(\theta)^2 - N(\theta) \cdot 2D(\theta)D'(\theta) = 0$. Since $D(\theta)$ is generally non-zero, this simplifies to the core condition:

$$N'(\theta)D(\theta) - 2N(\theta)D'(\theta) = 0 \tag{8}$$

The derivatives are: $N'(\theta) = -2k_{c2} \sin(2\theta) + 2k_{c3} \cos(2\theta)$ and $D'(\theta) = -2k_{m2} \sin(2\theta) + 2k_{m3} \cos(2\theta)$.

**Step 2: Convert to a Polynomial.** Let $x = \cos(2\theta)$ and $y = \sin(2\theta)$. Substituting into Eq. 8 gives $(-2k_{c2}y + 2k_{c3}x)(k_{m1} + k_{m2}x + k_{m3}y) - 2(k_{c1} + k_{c2}x + k_{c3}y)(-2k_{m2}y + 2k_{m3}x) = 0$. Expanding this equation yields a polynomial in $x$ and $y$ with terms up to degree 2 (e.g., $x^2, y^2, xy$). We rearrange this equation to isolate the terms containing $y$ $A_2(x)y^2 + A_1(x)y = A_0(x)$ where $A_i(x)$ are polynomials in $x$ of at most degree 2. We can now use the identity $y^2 = 1 - x^2$ to eliminate $y^2$. To eliminate the remaining $y$ term, we isolate it and square both sides of the equation. This process results in a final equation that is a polynomial in $x$ only. The highest power of $x$ will be 4, thus yielding a **quartic polynomial** in $x = \cos(2\theta)$.

## 5 Hierarchical Encoding via Reversible Merging (HERM)

Having established tools for pairwise analysis, we introduce the Hierarchical Encoding via Reversible Merging (HERM) framework to extend this analysis to an entire network. The HERM encoder applies a sequence of **algebraically reversible** operations to iteratively merge neuron pairs. This process transforms the network into two components: a compressed **skeletal** network, which is a minimal architecture where no more merging is possible, and a corresponding sequence of **recovery instructions**. The result is a meaningful hierarchical decomposition that can be perfectly reversed to restore the original model without any loss of information.

### 5.1 Construction of the Merged Neuron

The HERM merge operation is predicated on the GID factorization. Given a neuron pair $(i, j)$ from the same layer, we first compute their shared primitives $\{\mathbf{u}, \mathbf{v}, \mathbf{z}, \mathbf{r}\}$. Using these, we can then obtain neuron-specific reconstruction coefficients $(\alpha_k, \beta_k)_{k\in\{i,j\}}$. by solving the linear equality constraint from (2): $\mathbf{w}_k = \begin{bmatrix} \alpha_k \mathbf{u} + \beta_k \langle \mathbf{v}, \mathbf{r} \rangle \mathbf{z} \\ \beta_k \mathbf{v} + \alpha_k \langle \mathbf{u}, \mathbf{z} \rangle \mathbf{r} \end{bmatrix}$ for $k \in \{i, j\}$. This decomposition represents each neuron's weight vector $\mathbf{w}_k$ as the sum of a first-order term (e.g. $\alpha_k \mathbf{u}$) and a higher-order term (e.g., $\beta_k \langle \mathbf{v}, \mathbf{r} \rangle \mathbf{z}$) that models the pair's conditional logic. The merge operation then approximates the pair $(i, j)$ by replacing it with a new neuron, $\mathbf{w}_{\text{new}}$, constructed only from the first-order **primary** vectors $\mathbf{u}, \mathbf{v}$ and two new, as-yet-undetermined merge coefficients, $\alpha_0$ and $\beta_0$. This operation discards the higher-order interaction terms from $\mathbf{w}_{\text{new}}$, effectively creating a first-order approximation of the original pair:$\mathbf{w}_{\text{new}} \triangleq \begin{bmatrix} \mathbf{w}_{\text{new}}^{\text{in}} \\ \mathbf{w}_{\text{new}}^{\text{out}} \end{bmatrix} = \begin{bmatrix} \alpha_0 \mathbf{u} \\ \beta_0 \mathbf{v} \end{bmatrix}$.

### 5.2 The Ideal Cost of a Single Merge

The encoder greedily merges the neuron pair $(i, j)$ (from the same layer) that induces the **minimal distortion**. We define this distortion as the $L_2$-norm of the change in the block's output after the merge. The merge operation replaces the neuron $\mathbf{w}_i$ with a new, merged neuron $\mathbf{w}_{\text{new}}$, and eliminates $\mathbf{w}_j$ by zeroing its weights. Formally, let $\mathbf{x} \in \mathbb{R}^n$ be the block's input, $f : \mathbb{R}^n \to \mathbb{R}^c$ be its output function, and $\mathbf{w}_{\text{rest}}$ be all other weights. We assume a standard network layer whi $h$ neurons where $f(\mathbf{x}\,;\, \mathbf{w}_i, \mathbf{w}_j, \mathbf{w}_{\text{rest}}) = \sum_{k=1}^{h} \phi\left(\left\langle \mathbf{x}, \mathbf{w}_k^{\text{in}} \right\rangle\right) \mathbf{w}_k^{\text{out}} + g(\mathbf{x}; \mathbf{w}_{\text{rest}})$, where $g$ represents the function of the rest of the block and $\phi$ is the activation nonlinearity. We define the distortion as:

$$C_{\text{ideal}}(i,j) \triangleq \left\| f(\mathbf{x}\,;\, \mathbf{w}_i, \mathbf{w}_j, \mathbf{w}_{\text{rest}}) - f(\mathbf{x}\,;\, \mathbf{w}_{\text{new}}, \mathbf{0}, \mathbf{w}_{\text{rest}}) \right\| \tag{9}$$

$$= \left\| \sum_{k\in\{i,j\}} \phi\left(\left\langle \mathbf{x}, \mathbf{w}_k^{\text{in}} \right\rangle\right) \mathbf{w}_k^{\text{out}} - \phi\left(\langle \mathbf{x}, \alpha_0 \mathbf{u} \rangle\right) \beta_0 \mathbf{v} - \phi\left(\langle \mathbf{x}, \mathbf{0} \rangle\right) \mathbf{0} \right\|. \tag{10}$$

### 5.3 Input-Agnostic Coefficient Selection

The ideal cost $C_{\text{ideal}}(i, j)$ depends on the yet-to-be-determined coefficients $\alpha_0$ and $\beta_0$. A natural choice would be to select $\alpha_0, \beta_0$ that minimize $C_{\text{ideal}}$, but this optimization over a non-linear function $\phi$ is challenging. Instead, we minimize a proxy cost function $C_{\text{func}}(i, j)$, derived by approximating the activation with a linear function $\phi(x) \approx Ax + B$.

$$C_{\text{ideal}}(i,j) \;\approx\; C_{\text{func}}(i,j) \;\triangleq\; \left\| \left(A\left\langle \mathbf{x}, \mathbf{w}_i^{\text{in}} \right\rangle + B\right) \mathbf{w}_i^{\text{out}} + \left(A\left\langle \mathbf{x}, \mathbf{w}_j^{\text{in}} \right\rangle + B\right) \mathbf{w}_j^{\text{out}} - \left(A\left\langle \mathbf{x}, \alpha_0 \mathbf{u} \right\rangle + B\right) \beta_0 \mathbf{v} \right\| \tag{11}$$

The selection of optimal $\alpha_0, \beta_0$ must be input-agnostic, as $\mathbf{x}$ is unknown. A direct minimization of $C_{\text{func}}(i, j)$ would yield coefficients that are functions of $\mathbf{x}$, which is not permissible for a static network modification. To satisfy this constraint, we instead minimize a tractable and $\mathbf{x}$-independent upper bound $C_{\text{func}}^{\text{up}}(i, j)$. Assuming a known maximum input norm, $\rho \triangleq \sup \|\mathbf{x}\|$, and applying the triangle and matrix norm inequalities, one can obtain the following bound:

$$C_{\text{func}}(i,j) \le C_{\text{func}}^{\text{up}}(i,j) \triangleq \rho\,|A| \left\| \mathbf{w}_i^{\text{out}} \mathbf{w}_i^{\text{in}\top} + \mathbf{w}_j^{\text{out}} \mathbf{w}_j^{\text{in}\top} - \alpha_0\,\beta_0\,\mathbf{v}\mathbf{u}^\top \right\|_F + |B| \left\| \mathbf{w}_i^{\text{out}} + \mathbf{w}_j^{\text{out}} - \beta_0 \mathbf{v} \right\|. \tag{12}$$

This derivation yields an elegant and practical solution with two crucial properties. First, although the upper bound itself depends on the unknown linearization constants $A, B$ and the input norm $\rho$, the optimal coefficients $\alpha_0^*, \beta_0^*$ that minimize it are, remarkably, independent of all three. Second, the solution is symmetric with respect to the neuron indices $(i, j)$. This symmetry provides a computational advantage: while the cost functions $C_{\text{ideal}}(i, j)$ and $C_{\text{func}}(i, j)$ are asymmetric, the underlying optimal coefficients $(\alpha_0^*, \beta_0^*)$ are identical for both merge directions, precluding the need for a redundant calculation. It is straightforward to derive the expressions for $\alpha_0^*, \beta_0^*$ that minimize $C_{\text{func}}^{\text{up}}(i, j)$, which have the following form:

$$\alpha_0^* = \frac{\left\langle \mathbf{u}, \mathbf{w}_i^{\text{in}} \right\rangle \left\langle \mathbf{v}, \mathbf{w}_i^{\text{out}} \right\rangle + \left\langle \mathbf{u}, \mathbf{w}_j^{\text{in}} \right\rangle \left\langle \mathbf{v}, \mathbf{w}_j^{\text{out}} \right\rangle}{\left\langle \mathbf{v}, \mathbf{w}_i^{\text{out}} + \mathbf{w}_j^{\text{out}} \right\rangle},\; \beta_0^* = \left\langle \mathbf{v}, \mathbf{w}_i^{\text{out}} + \mathbf{w}_j^{\text{out}} \right\rangle \tag{13}$$

Note that the above expression for $\alpha_0^*$ is well-defined when $\left\langle \mathbf{v}, \mathbf{w}_i^{\text{out}} + \mathbf{w}_j^{\text{out}} \right\rangle \neq 0$. If $\mathbf{v}$ is orthogonal to $\mathbf{w}_i^{\text{out}} + \mathbf{w}_j^{\text{out}}$ this condition is not met, but then the optimal $\beta_0$ must be zero $\beta_0^* = 0$, which means $C_{\text{ideal}}(i, j)$ becomes independent of $\alpha_0$. Therefore, in this degenerate case, $\alpha_0^*$ is arbitrary, and we can set it to 0 by convention.

### 5.4 Pair Selection

With the optimal coefficients $(\alpha_0^*, \beta_0^*)$ computed for a pair $(i, j)$, the merged neuron $\mathbf{w}_{\text{new}}^*$ is fully defined. We must now select which pair to merge by evaluating this potential merge with a cost function. Both $C_{\text{ideal}}(i, j)$ and its proxy $C_{\text{func}}(i, j)$ are unsuitable for this selection process, as they are input-dependent (relying on $\mathbf{x}$ and its norm $\rho$). We therefore introduce a third cost function $C_{\text{struct}}(i, j)$, which is a purely **structural proxy** that is fully input-agnostic. This cost approximates the functional distortion by measuring the total perturbation to the network's weight space. This perturbation is defined as the sum of the squared $L_2$ norms of the change vectors for the survivor neuron ($i$) and the victim neuron ($j$):

$$C_{\text{struct}}(i,j) \triangleq \underbrace{\left\|\mathbf{w}_{\text{new}}^* - \mathbf{w}_i\right\|^2}_{\text{Survivor Perturbation}} + \underbrace{\left\|\mathbf{0} - \mathbf{w}_j\right\|^2}_{\text{Victim Perturbation}} = \left\|\mathbf{w}_{\text{new}}^* - \mathbf{w}_i\right\|^2 + \left\|\mathbf{w}_j\right\|^2 \tag{14}$$

This structural cost $C_{\text{struct}}$ inherits the asymmetry of $C_{\text{ideal}}$, as $C_{\text{struct}}(i, j) \neq C_{\text{struct}}(j, i)$. To select a single optimal pair $(i^*, j^*)$, we must aggregate these two asymmetric costs into a single, symmetric cost value for the pair. While the minimum of the two could be used, we proceed with their average, as it provides a more robust and conservative estimate of the pair's mutual interchangeability. This defines our selection criterion: $(i^*, j^*) = \text{argmin}_{(i,j)} \frac{1}{2}\left(C_{\text{struct}}(i, j) + C_{\text{struct}}(j, i)\right)$.

### 5.5 Survivor and Victim Assignment

Once the optimal pair $(i^*, j^*)$ in the network is decided, we should decide which of these two neurons should be the survivor (to be replaced by $\mathbf{w}_{\text{new}}^*$) and which one should be the victim (to be replaced by $\mathbf{0}$) from the winning pair $(i^*, j^*)$. A rational choice is to select the direction that causes the least possible damage according to $C_{\text{struct}}(i, j)$. That is:

$$(\mathbf{w}_{\text{survivor}}, \mathbf{w}_{\text{victim}}) = \begin{cases} (\mathbf{w}_{i^*}, \mathbf{w}_{j^*}) & \text{if } C_{\text{struct}}(i^*, j^*) \le C_{\text{struct}}(j^*, i^*) \\ (\mathbf{w}_{j^*}, \mathbf{w}_{i^*}) & \text{otherwise} \end{cases}$$

The network is then modified by applying the update: $\mathbf{w}_{\text{survivor}} \leftarrow \mathbf{w}_{\text{new}}^*$ and $\mathbf{w}_{\text{victim}} \leftarrow \mathbf{0}$.

### 5.6 Recovery Instructions

The merge operation that replaces the pair $(i, j)$ with $\mathbf{w}^*_{\text{new}}$ is a lossy, first-order approximation that discards the higher-order interaction terms. To ensure the entire encoding process is perfectly reversible, the algorithm must archive all information necessary to reconstruct the *original* $\mathbf{w}_i$ and $\mathbf{w}_j$ from scratch. For each merge, a **recovery instruction packet**, $\mathcal{I}_{ij}$, is created and stored. This packet contains the complete set of parameters that define the original pair's 2D subspace and their exact coordinates within it: the location of the merged pair (including the layer index): $(i, j, \ell)$, The shared GID primitives: $\{\mathbf{u}, \mathbf{v}, \mathbf{z}, \mathbf{r}\}$, The original reconstruction coefficients: $\{\alpha_i, \beta_i, \alpha_j, \beta_j\}$. As defined in the GID formulation (2), these components are mathematically sufficient for the decoder to perfectly reconstruct the original $\mathbf{w}_i$ and $\mathbf{w}_j$ via direct algebraic synthesis.

### 5.7 Complete Encoding and Decoding

The HERM framework is composed of two algorithms: an encoder that iteratively compresses the network, and a decoder that perfectly reverses the process.

#### 5.7.1 Encoder Algorithm

The encoder transforms the full network into a skeletal model and a LIFO (Last-In, First-Out) queue of recovery instructions, $\mathcal{I}$. It iteratively applies the following four-step cycle until no more valid pairs can be merged: **Pair Selection:** Find the optimal pair $(i^*, j^*)$ in the entire network by solving the global minimization problem using our symmetric structural proxy $(i^*, j^*) = \text{argmin}_{(i,j)} \frac{1}{2} \left(C_{\text{struct}}(i, j) + C_{\text{struct}}(j, i)\right)$. **Direction Assignment:** Determine the optimal merge direction by finding the minimum of $C_{\text{struct}}(i^*, j^*)$ and $C_{\text{struct}}(j^*, i^*)$. Compute $\mathbf{w}^*_{\text{new}}$ for this optimal assignment. **Store Instructions:** Archive the complete recovery instruction packet $\mathcal{I}_{i^*j^*}$ (containing the GID primitives and original coefficients $\{\alpha_{i^*}, \beta_{i^*}, \alpha_{j^*}, \beta_{j^*}\}$) by pushing it onto the instruction queue $\mathcal{I}$. **Network Update:** Simplify the network by setting the designated victim neuron's weights to $\mathbf{0}$ and replacing the survivor neuron's weights with the computed $\mathbf{w}^*_{\text{new}}$.

#### 5.7.2 Decoder Algorithm

The decoder perfectly reconstructs the original network by reversing this process. Starting from the final skeletal model, it iteratively applies the instructions in the reverse order of their creation, systematically executing an **un-merging** operation at each step until the network is restored to its full state. Each un-merging step is a deterministic algebraic substitution that perfectly restores two "child" neurons from a single "parent" neuron, demonstrating that the encoding is fully reversible. Specifically, the decoder repeats the following steps until the instruction queue $\mathcal{I}$ is empty:

1. **Read Instruction:** Pop the next instruction packet $\mathcal{I}_{ij}$ from the queue.
2. **Expand Network:** Re-allocate space for two neurons at locations $(i, j)$, replacing the single merged neuron that was stored at the survivor's location.
3. **Reconstruct Weights:** Using the GID primitives and reconstruction coefficients from the instruction packet, perfectly synthesize the original weight vectors $\mathbf{w}_i$ and $\mathbf{w}_j$ via the GID synthesis equation (2).

### 5.8 Rationale for the Hybrid Proxy Strategy

A reader might wonder about the necessity of this hybrid-proxy approach: using $C^{\text{up}}_{\text{func}}$ to derive the coefficients $(\alpha^*_0, \beta^*_0)$, but then using $C_{\text{struct}}$ to select the optimal pair $(i^*, j^*)$. This is not a redundancy but a deliberate design choice that matches the complexity of each proxy to the specific requirements of its task.

For **coefficient derivation**, $C^{\text{up}}_{\text{func}}$ is ideal because while it depends on unknown scalars $(\rho, A, B)$, these do not change the *location* of the minimum. We can therefore find the optimal coefficients $(\alpha^*_0, \beta^*_0)$ using this high-fidelity proxy. Using the "weaker" $C_{\text{struct}}$ here would yield poorer approximation quality.

The **pair selection** is a comparison problem. Here, the $\rho, A, B$ scalars in $C^{\text{up}}_{\text{func}}$ matter and affect the choice of $(i, j)$ but they are unknown. By contrast, while $C_{\text{struct}}$ is a lower-fidelity approximation of $C_{\text{ideal}}$, it doesn't rely on unknown quantities and can be directly used for comparing different neuron pairs $(i, j)$.

### 5.9 Practical Consideration

**Optional Residual Correction for Perfect Reconstruction.** While the HERM framework is algebraically lossless, the GID factorization relies on arithmetic operations that are subject to finite floating-point precision. This can result in **minuscule round-off errors**, where the reconstructed weights $\mathbf{w}^{\text{recon}}_k$ are high-fidelity approximations but not bit-for-bit identical to the original $\mathbf{w}_k$. For applications requiring provable, perfect reconstruction, a simple **residual correction** mechanism can be enabled. This add-on involves the encoder calculating the numerical residual vector, $\Delta\mathbf{w}_k \triangleq \mathbf{w}_k - \mathbf{w}^{\text{recon}}_k$, and storing it in the instruction packet $\mathcal{I}_{ij}$. The decoder then adds this stored residual back during synthesis ($\mathbf{w}^{\text{final}}_k = \mathbf{w}^{\text{recon}}_k + \Delta\mathbf{w}_k$), guaranteeing bit-for-bit reversibility. For application scenarios studied Section 6, perfect bit-for-bit fidelity is not required. However, we present the full details of this optional mechanism in the Appendix B for completeness and for future applications where provable, exact reversibility is required.

**Handling Normalization Layers.** For clarity of presentation, our main development in the text assumed a simplified network architecture without normalization. In practice, modern architectures, including all models evaluated in our experiments in Section 6, rely on Group Normalization (GN). This introduces a non-trivial but critical implementation detail, as the learnable affine parameters $(\gamma, \delta)$ of the GN layers must be systematically handled to ensure the algebraic reversibility of the encode/decode process. This requires a specific **asymmetric folding** strategy for constructing the joint weight vectors $(\mathbf{w}_n, \mathbf{w}_c)$ and principled rules for merging and restoring the GN parameters. As this mechanism is a practical extension rather than part of the core theory, we defer its full derivation and implementation details to the Appendix A.

**Adaptation for Convolutional Layers** For clarity of presentation, our main development in the text assumed a fully connected network architecture. However, we can easily adapt our results to work with convolutional networks by defining a "neuron" as a single filter $i$ within a layer `conv_a`. The joint vector $\mathbf{w}_i = [\mathbf{w}_n; \mathbf{w}_c]$ for this filter is constructed by vectorizing its input and output weights. The **n-space** $\mathbf{w}_n$ is simply the flattened weight kernel of the

filter. If `conv_a` has a kernel $K_A$ of shape $(h_A, w_A, C_{\text{in}}, C_{\text{out}})$, then $\mathbf{w}_n$ for filter $i$ is the flattened vector $K_A[:, :, :, i]$, with dimension $n = h_A \times w_A \times C_{\text{in}}$. The **c-space** $\mathbf{w}_c$ represents the filter's influence on the subsequent layer, `conv_b`, which has kernel $K_B$. Since the output map of filter $i$ becomes the $i$-th input channel to `conv_b`, $\mathbf{w}_c$ is constructed by collecting and flattening all weights in $K_B$ that operate on this $i$-th input channel (i.e., $K_B[:, :, i, :]$). This results in a vector of dimension $c = h_B \times w_B \times C_{\text{out},B}$. This representation allows the GID factorization and all subsequent merge operations to be applied to convolutional layers directly. In fact, all the experiments in Section 6 use a convolutional network.

We present the complete pseudocode for the HERM encoder and decoder in Appendix C.3. **These algorithms integrate all the practical considerations discussed above**.

# 6 Proof-of-Concept Applications

All experiments are conducted on the CIFAR-10 dataset. We use a simple convolutional neural network for all of the experiments defined by the sequence: `Conv(32)` → `GroupNorm` → `MaxPool` → `Conv(64)` → `GroupNorm` → `MaxPool` → `Flatten` → `Dense(128)` → `GroupNorm` → `Dense(64)` → `GroupNorm` → `Dense(10)`. The two convolutional layers use 3x3 kernels. This architecture relies on `GroupNormalization` after each main layer.

## 6.1 Application 1: Dispersed Elastic Fine-Tuning (DEFT)

HERM provides a principled way to guide fine-tuning by distinguishing fundamental concepts (merged late, inelastic) from granular specializations (merged early, elastic).

### 6.1.1 Methodology

**Dispersed Elastic Fine-Tuning (DEFT)** leverages the hierarchy from a HERM encoding to control parameter plasticity during transfer learning.

1. **Elasticity Map Generation:** A one-time HERM encoding is performed on the source model, producing a decoding curriculum of $M$ instructions. An **elasticity map** $\mathcal{C}$—a set of tensors matching the model's parameter shapes—is initialized to zeros.
2. **Map Population:** The curriculum is traversed. For each un-merge instruction $I_m$ (at step $m$), a hierarchical elasticity $e_m = m/M$ is assigned. The GID primitives from $I_m$ are used to reconstruct the *interactional components* of the un-merged neurons' weights (e.g., $(\mathbf{w}_{i,\text{int}})_n = \beta_i \langle \mathbf{v}, \mathbf{r} \rangle \mathbf{z}$). These components represent the specialized, conditional logic. The elasticity map $\mathcal{C}$ is then updated at the corresponding parameter locations by taking the maximum of the current value and the normalized magnitude of this interactional component, scaled by $e_m$.
3. **Propagation:** To maintain functional coherence, a neuron's final elasticity (the max value from its kernel weights) is broadcast to all of its associated parameters, including its bias and Group Normalization (beta, gamma) parameters.
4. **Gradient Modulation:** The downstream fine-tuning process then uses this map $\mathcal{C}$ to modulate the gradient update for each parameter tensor $\boldsymbol{\theta}_k$ (identified by its path, $\text{path}_k$): $\Delta\boldsymbol{\theta}_k \triangleq -\eta \cdot \mathcal{C}[\text{path}_k] \odot \frac{\partial L_D}{\partial \boldsymbol{\theta}_k}$. This update rule freezes core parameters (where $\mathcal{C} \approx 0$) while allowing full adaptation of specialized ones (where $\mathcal{C} \approx 1$).

### 6.1.2 Experimental Validation

DEFT was compared against a standard frozen-backbone across 20 transfer learning tasks (binary classifications derived from CIFAR-10). DEFT achieved higher accuracy in **17 of the 20** scenarios, with significant gains (e.g., $+11.9\%$, $+13.4\%$), validating that a principled, dispersed plasticity is superior to coarse-grained freezing.

## 6.2 Application 2: Self-Organizing Architectures

The algebraic reversibility of HERM operators provides the mechanics for networks to dynamically adapt their own architecture during training.

### 6.2.1 Methodology

An experiment was run where a network training on CIFAR-10 is periodically reconfigured by a controller that can **Expand** or **Shrink** the architecture. The HERM/GID formalism provides the precise mechanics for these operations.

- **Expansion (Splitting):** Triggered by a plateau in validation accuracy. The target neuron $k^*$ for splitting is the one with the largest gradient L2-norm, $k^* = \text{argmax}_k \|\nabla_{\mathbf{w}_k} L\|_2$, ensuring capacity is added where needed. The expansion is the *algebraic inverse* of a HERM merge. The single neuron $k^*$ is replaced by a new pair $(i, j)$. Their new GID coefficients are initialized to ensure reversibility: input coefficients are duplicated ($\alpha_i = \alpha_j = \|(\mathbf{w}_{k^*})_n\|$), while output coefficients are halved ($\beta_i = \beta_j = \frac{1}{2}\|(\mathbf{w}_{k^*})_c\|$).
- **Shrinking (Merging):** Triggered by signs of over-parameterization. The network uses the HERM encoder's logic to find the pair of neurons $(i^*, j^*)$ with the minimal **True Structural Cost**, $C_{\text{struct}}(i, j) \triangleq \|\mathbf{w}_{\text{new}} - \mathbf{w}_i\|^2 + \|\mathbf{w}_j\|^2$, where $\mathbf{w}_{\text{new}}$ is the GID-based merged prototype. This identifies the pair that can be merged with the least functional disruption, making it the safest candidate for removal.

### 6.2.2 Results and Discussion

The results in Figure 4 clearly show dynamic architectural adaptation. The network autonomously allocated more neurons to the first convolutional layer and final bottleneck layer while slightly pruning a dense layer, intelligently reallocating its parametric resources. While architectural changes introduce transient instabilities in test accuracy, the model demonstrates the ability to recover and continue learning. This serves as a compelling proof-of-concept that the GID-based operators function as designed, enabling a principled approach to self-organizing network architectures.

## 6.3 Application 3: Model Compression

Because the decoder reconstructs the full model by progressively expanding a skeletal one, a partially decoded model can be treated as a compressed version of the original. This allows a full model to be encoded once, after which partially decoded versions at various stages can be generated inexpensively. These versions can then serve different applications, each requiring a specific trade-off between compression and fidelity.

We use structured $L_1$ pruning as a baseline compression method. This approach prunes entire neurons (effectively removing all associated incoming and outgoing weights) rather than individual parameters, ensuring a more equitable

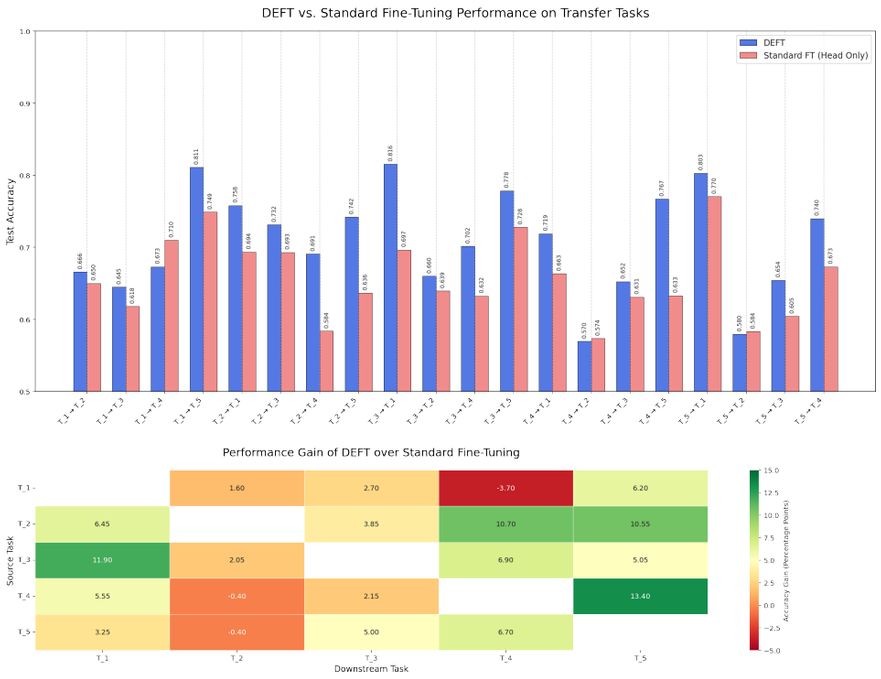


Figure 3: Visual comparison of DEFT and Standard fine-tuning performance. **(Top)** A grouped bar chart directly compares the final test accuracy for all 20 transfer-learning tasks. For each pair, the DEFT result (blue) is shown next to the Standard baseline (red), making the performance difference clear. **(Bottom)** A heatmap visualizes the performance gain of DEFT over the baseline, measured in percentage points. Green cells indicate a performance increase with DEFT, while red indicates a decrease. The strong prevalence of green cells visually confirms DEFT's consistent and significant advantage across the majority of transfer tasks.

comparison against HERM. A global threshold $\tau$ is set, and any neuron $j$ whose associated weights have an $L_1$ norm less than $\tau$ is pruned.

Our experiments evaluated the relationship between model density (the ratio of active parameters to the total number of parameters) and test set accuracy. As shown in Figure 4, HERM yields more accurate models than $L_1$ pruning in higher-density regimes. However, this trend reverses for very sparse (low-density) models.

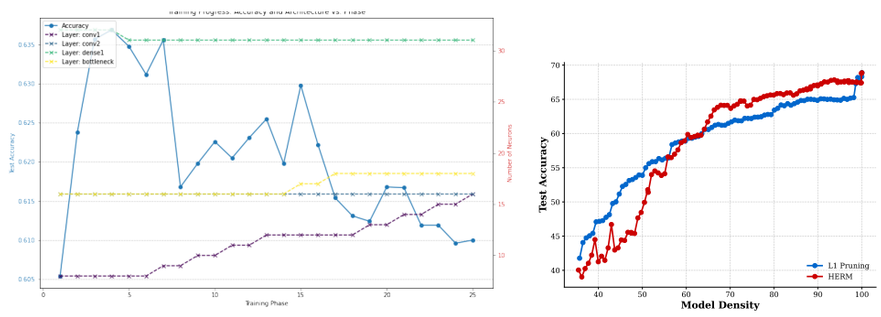


Figure 4: **Left.** Training progress over 25 phases. The primary y-axis (left, blue) shows the model's test accuracy. The secondary y-axis (right, red) tracks the number of neurons in each of the four dynamically sized layers, demonstrating the network's ability to autonomously reallocate its parameters. **Right.** Model density versus accuracy plot for structured L1 pruning and HERM.

## 7 Conclusion

In this work, we have presented a multi-level mathematical framework for the analysis and representation of neural networks. We began by developing two mathematically-grounded, general-purpose tools for pairwise neuron analysis: Structural Component Analysis (SCA), which finds a maximally structure-aligned orthonormal basis, and the Gated Interaction Decomposition (GID), which provides an interpretable model of the coupling mechanism between a neuron's input and output weights.

Building on these tools, we introduced the Hierarchical Encoding via Reversible Merging (HERM) framework. By using GID as a provably lossless algebraic operator, HERM refactors an entire trained network into a skeletal form and a set of recovery instructions, revealing a meaningful functional hierarchy of its learned knowledge. We positioned this framework not as an empirical method, but as a foundational toolkit. As a proof of concept, we demonstrated its utility in two distinct applications: guiding a principled elastic fine-tuning strategy and enabling the creation of dynamic, self-organizing architectures.

The primary contribution of this work is the framework itself. By providing tools that respect neural structure and uncover hierarchical relationships, SCA, GID, and HERM offer a new and lens through which to understand, interpret, and manipulate deep neural networks. Future work could explore the use of this framework for more efficient model compression, targeted knowledge transfer, and deeper interpretability studies.

## REFERENCES


E. Adly and et al. Uncovering millions of features in claude 3 sonnet. *Anthropic Research*, May 2024.

Michal Aharon, Michael Elad, and Alfred Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Quoc Le, and Sashank Reddi. Gqa: Training generalized rpe transformers for gqa. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Galen Andrew, Raman Arora, Jeff A Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International Conference on Machine Learning (ICML)*, pp. 1247–1255. PMLR, 2013.

Horace B Barlow. Possible principles underlying the transformation of sensory messages. *Sensory Communication*, 1: 217–234, 1961.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6541–6549, 2017.

Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience*, 10(3):186–198, 2009.

György Buzsáki. Neural syntax: cell assemblies, cognition, and behavior. *Annu. Rev. Neurosci.*, 33:1–24, 2010.

Pierre Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314, 1994.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Daniel J Felleman and David C Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1(1):1–47, 1991.

Albert Gidon, Talfan A Zolnik, Jürgen Falter, Athanasia Papoutsi, Panayiota Poirazi, Wulfram Rössler, Johannes Schiess, Dietmar Schmitz, Nelson Spruston, and Matthew E Larkum. Dendritic action potentials in human neocortical neurons. *Science*, 367(6473):83–87, 2020.

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

David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. Kernel canonical correlation analysis. *Journal of Machine Learning Research*, 5:815–836, 2004.

Richard A Harshman. Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

Michael Häusser and Jeffrey C Magee. Dendritic computation: a new perspective on the neuron. *Nature Reviews Neuroscience*, 22(8):475–487, 2021.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

A. Holtmaat et al. Formation and maintenance of cortical memory engrams. *Nature Reviews Neuroscience*, 23(4): 220–234, 2022.

Emiel Hoogeboom, Rianne van den Berg, Sam Bond-Taylor, and Max Welling. The subspace structure of neural representations. *arXiv preprint arXiv:2310.03324*, 2023.

Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933.

Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.

Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

Simon Kornblith, Mohammad Norouzi, Hanieh Hong, and Geoffrey Lee. Similarity of neural network representations revisited. In *International Conference on Machine Learning (ICML)*, pp. 3519–3529. PMLR, 2019.

Jack Lanchantin, Chandar Singh, Tom Goldblum, and Matthew Johnson. Sparse autoencoders for scalable mechanistic interpretability. In *NeurIPS 2023 Workshop on Mechanistic Interpretability*, 2023.

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401 (6755):788–791, 1999.

Michael London and Michael Häusser. Dendritic computation. *Annu. Rev. Neurosci.*, 28:503–532, 2005.

Bartlett W Mel. Information processing in dendritic trees. *Neural Computation*, 6(6):1031–1085, 1994.

Neel Nanda, Evan Michaud, and Nelson Liu. Progress in mechanistic interpretability. *Distill*, 5(1):e100003, 2023.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e0001a–zoomin, 2020.

Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997.

Karl Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

R. S. Pimentel, J. Roncalli, C. B. Peterson, and D. B. Dunson. Data integration via analysis of subspaces (divas). *arXiv preprint arXiv:2212.00703*, 2024.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear networks. *arXiv preprint arXiv:1312.6120*, 2013.

A. Shapson-Coe et al. A petascale automated connectome of the human temporal lobe. *Nature*, 631:1–8, 2024.

Olaf Sporns. *Networks of the brain*. MIT press, 2010.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 5998–6008, 2017.

Daniela M Witten, Robert Tibshirani, and Trevor Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534, 2009.

Herman Wold. Partial least squares. In *Encyclopedia of Statistical Sciences*. Wiley, 1975.

# APPENDIX CONTENTS

# A HANDLING GROUP NORMALIZATION

This section details the specific strategies used in the code to handle the learnable scale $\gamma$ and shift $\delta$ parameters of GN layers during the encoding and decoding processes.

## A.1 ENCODER: ASYMMETRIC FOLDING FOR JOINT WEIGHT CONSTRUCTION

A key step is constructing the joint weight vector $\mathbf{w} = [\mathbf{w}_n; \mathbf{w}_c]$ for a neuron in a layer `A` that is followed by a GN layer (`GN-A`) and then another convolutional/dense layer `B`.

- **The n-space ($\mathbf{w}_n$):** The input-space vector is constructed from the **effective weights** of layer 'A'. This is achieved by "folding" the channel-wise scaling parameter, $\boldsymbol{\gamma}_A$, from 'GN-A' into the bare kernel weights of layer 'A'.
$$(\mathbf{w}_i)_n \longleftarrow \text{flatten}(\text{kernel}_A[:, :, :, i] \cdot \gamma_A[i]) \tag{15}$$
This correctly represents the actual linear operation performed by neuron $i$ on its inputs.
- **The c-space ($\mathbf{w}_c$):** Crucially, the output-space vector is constructed from the **bare weights** of the subsequent layer `B`. The code *does not* fold the $\boldsymbol{\gamma}_B$ from the next GN layer (`GN-B`). This asymmetric handling is a deliberate and necessary correction. The $\boldsymbol{\gamma}_B$ parameter scales the *outputs* of layer 'B', not its inputs. Therefore, to correctly model the influence of neuron $i$ from layer 'A' *on* layer `B`, we must use the bare weights of `B` that operate on the output of neuron $i$.

## A.2 ENCODER: PRINCIPLED MERGING OF GN PARAMETERS

Before a pair of neurons $(i, j)$ can be merged, it must pass the three-part numerical invertibility check on its $\gamma$ parameters, ensuring that both the encoder and decoder can perform the necessary inverse operations without encountering a division by zero. Only after a pair is deemed eligible are its corresponding GN parameters merged. The merge rules for GN parameters are chosen to be simple, axiomatic approximations as presented below.

- **Merging Gamma ($\gamma$):** The new parent neuron's gamma is the **average** of the child gammas, preserving the mean activation sensitivity
$$\gamma_{\text{parent}} = \frac{1}{2}(\gamma_i + \gamma_j) \tag{16}$$
After the effective n-space vector of the parent, $\mathbf{w}^{\text{eff}}_{\text{parent},n}$, is computed, it must be "unfolded" using this new gamma to find the bare kernel weight to be stored in the network: $\text{kernel}_{\text{parent}} = \mathbf{w}^{\text{eff}}_{\text{parent},n}/\gamma_{\text{parent}}$.
- **Merging Delta ($\delta$):** The new parent neuron's delta (the bias/shift term) is the **sum** of the child deltas, preserving the total additive bias passed to the next layer.
$$\delta_{\text{parent}} = \delta_i + \delta_j \tag{17}$$

The asymmetric merge rules for the Group Normalization parameters are derived from the same axiomatic principles that govern the GID coefficients. The scaling parameter, $\gamma$, is a multiplicative factor that controls the sensitivity of the neuron's activation. To best preserve the pre-activation statistics, the new parent neuron's $\gamma$ is defined as the **average** of the child gammas ($\gamma_{\text{new}} = \frac{1}{2}(\gamma_i + \gamma_j)$). In contrast, the shift parameter, $\delta$, is a pure additive bias. To preserve the total additive signal passed to the subsequent layer, its value is defined as the **sum** of the child deltas ($\delta_{\text{new}} = \delta_i + \delta_j$).

## A.3 DECODER: REVERSING THE PROCESS

The decoder perfectly reverses these operations. For each un-merge instruction:

1. The original $\gamma_i, \delta_i, \gamma_j, \delta_j$ values for the child neurons are retrieved from the instruction packet.
2. The GID primitives are used to reconstruct the final *effective* child weights, $\{\mathbf{w}^{\text{final}}_i, \mathbf{w}^{\text{final}}_j\}$.
3. The n-space components of these effective weights are "unfolded" by dividing by their original, respective gamma values ($\mathbf{w}^{\text{bare}}_{i,n} = \mathbf{w}^{\text{final}}_{i,n}/\gamma_i$). The c-space components are already bare and require no unfolding.
4. The bare kernel weights are placed back into the network layers, and the original $\gamma$ and $\delta$ values are restored in the GN layer, ensuring a perfect reconstruction.

## A.4 ENSURING NUMERICAL STABILITY IN THE MERGE OPERATION

A critical aspect of numerical stability involves handling neurons with near-zero magnitudes. For such neurons, the matrices derived from their weights become ill-conditioned, rendering the main GID factorization algorithm numerically unstable. To handle these "trivial merge" cases gracefully, the implementation includes a bypass mechanism. If both neurons in a candidate pair have a norm below a small tolerance, the GID solver is avoided entirely. The merge cost is computed directly from the norms of the original neurons ($C_{\text{struct}} \triangleq \|\mathbf{w}_i\|^2 + \|\mathbf{w}_j\|^2$), and the GID primitives are treated as zero vectors, as there is no meaningful interaction to model. This practical optimization ensures robustness without compromising the integrity of the cost function.

A central requirement of the hierarchical coding scheme is its perfect, lossless reversibility. This implies that the full encoding process, viewed as a map from a pair of neurons to a merged neuron and a recovery instruction, must be perfectly invertible. While the GID is algebraically exact, its practical implementation within a network containing Group Normalization (GN) layers introduces a critical precondition for this **invertibility**.

The potential point of failure arises from the forward operation of the GN layer, which scales a neuron's bare kernel weights, $\mathbf{w}_{\text{bare}}$, by a learnable parameter, $\boldsymbol{\gamma}$, to produce its effective weights: $\mathbf{w}_{\text{eff}} = \mathbf{w}_{\text{bare}} \cdot \boldsymbol{\gamma}$. Consequently, the decoder must perform the inverse operation, $\mathbf{w}_{\text{bare}} = \mathbf{w}_{\text{eff}}/\boldsymbol{\gamma}$, to restore the original network state. This division is mathematically undefined for $\boldsymbol{\gamma} = 0$ and numerically ill-conditioned for $|\boldsymbol{\gamma}| \approx 0$.

To guarantee a robust and reversible merge, the encoder must therefore only select pairs $(i, j)$ that satisfy the necessary invertibility conditions. This leads to a principled, three-part eligibility check, where $\epsilon$ is a small, positive constant for numerical tolerance:

1. **First Child Neuron Invertibility:** The scaling parameter for the first neuron in the pair must be numerically invertible. $|\gamma_i| \geq \epsilon$.
2. **Second Child Neuron Invertibility:** The scaling parameter for the second neuron in the pair must also be numerically invertible. $|\gamma_j| \geq \epsilon$.
3. **Parent Neuron Invertibility:** The scaling parameter for the newly created parent neuron, defined as $\gamma_{\text{new}} = \frac{1}{2}(\gamma_i + \gamma_j)$, must also be numerically invertible, as the encoder itself performs this inverse operation. $|\gamma_{\text{new}}| \geq \epsilon$.

A neuron pair is deemed **eligible** for a merge only if it satisfies all three of these conditions. This check is not a heuristic; it is a direct test of the mathematical preconditions for the operation's reversibility.

Implementing this check efficiently within the greedy search for the best pair can be approached with two distinct computational strategies:

- **Post-Check Strategy:** This iterative approach first performs an exhaustive search to find the single best candidate pair based on a primary cost metric (e.g., $C_{\text{struct}}$). Only then are the $\boldsymbol{\gamma}$ parameters of this single candidate subjected to the three-part eligibility check. If the check fails, the pair is added to an exclusion list, and the entire search is restarted. This minimizes expensive computations when top candidates are likely to be valid.
- **Unified Strategy:** This approach guarantees a **single-pass search**. It computes the GID factorization, the merge cost, and the result of the $\boldsymbol{\gamma}$ eligibility check for *all* possible pairs upfront, typically via a vectorized computation. It then finds the best pair that is certified as valid from this complete set of results. While computationally intensive upfront, it avoids the possibility of costly re-search loops, making it faster and more predictable in cases where many top-ranked candidates might be ineligible.

By combining a principled cost function with this rigorous, architecture-aware eligibility check, the encoder can reliably and efficiently select neuron pairs for a merge operation that is guaranteed to be perfectly and robustly reversible.

### A.5 Choice of Normalization Scheme

The presented mechanism for handling Group Normalization (GN) is not universally applicable to all normalization schemes; it is, in fact, uniquely suited to GN due to its "local" properties. The core HERM assumption is that a pairwise merge of neurons $(i, j)$ does not create side-effects that alter the output of other neurons in the layer. GN adheres to this, as its statistics are computed per-channel (or per-group), making each neuron's normalized output independent of its peers. Other schemes present fundamental challenges: Layer Normalization (LN) is incompatible, as it normalizes *across* the entire layer, meaning a local merge would change the statistics and thus the output of all other neurons. Batch Normalization (BN) presents a different difficulty, as it requires a new, non-trivial merge rule for its non-learnable statistical parameters (*moving_mean*, *moving_variance*), which our framework does not define.

## B Ensuring Numerical Reversibility via Residual Correction

A central claim of the hierarchical coding scheme is its perfect, lossless reversibility. While this holds true from a purely algebraic perspective, the practical implementation introduces a subtle challenge. The GID factorization for the general, two-sided interaction case relies on numerical optimization algorithms. These algorithms operate with finite floating-point precision and are subject to convergence tolerances, inevitably introducing minuscule numerical errors. Consequently, the GID primitives found by the encoder are high-quality numerical approximations, not exact algebraic solutions. When the decoder uses these slightly imprecise primitives, the reconstructed network will be nearly identical to the original but not bit-for-bit perfect, technically violating the lossless guarantee.

To bridge this gap between theoretical algebra and practical implementation, we introduce a simple and robust correction mechanism. The core idea is to compute, store, and re-apply the small numerical error, or **residual**, that arises during the encoding process. This ensures that any precision lost during factorization is perfectly accounted for during reconstruction.

### B.1 Encoder Modification: Archiving the Residual

The encoding algorithm is augmented with a final verification and correction step. After a neuron pair $(\mathbf{w}_i, \mathbf{w}_j)$ is selected and its GID factorization is numerically computed, the encoder performs the following:

1. **Immediate Reconstruction:** Using the just-computed numerical GID primitives and coefficients, the encoder immediately reconstructs the neuron weights, yielding a reconstructed pair $\{\mathbf{w}_i^{\text{recon}}, \mathbf{w}_j^{\text{recon}}\}$.
2. **Residual Calculation:** The encoder computes the small vector difference between the original and reconstructed weights. This difference is the numerical residual:

$$\Delta\mathbf{w}_i = \mathbf{w}_i - \mathbf{w}_i^{\text{recon}} \tag{18}$$

$$\Delta\mathbf{w}_j = \mathbf{w}_j - \mathbf{w}_j^{\text{recon}} \tag{19}$$

3. **Store Residual:** These two residual vectors, $\Delta\mathbf{w}_i$ and $\Delta\mathbf{w}_j$, are archived in the recovery instruction set $I_{ij}$ alongside the GID primitives and coefficients.

### B.2 Decoder Modification: Applying the Correction

The decoder's process is correspondingly updated with a final correction step to achieve perfect reconstruction:

1. **Reconstruction from Primitives:** The decoder uses the GID primitives from the recovery instructions to reconstruct the neuron weights, yielding the same $\{\mathbf{w}_i^{\text{recon}}, \mathbf{w}_j^{\text{recon}}\}$ as the encoder.

2. **Residual Correction:** The decoder retrieves the residual vectors $\Delta\mathbf{w}_i$ and $\Delta\mathbf{w}_j$ from the instructions and adds them back to the reconstructed weights:

$$\mathbf{w}_i^{\text{final}} = \mathbf{w}_i^{\text{recon}} + \Delta\mathbf{w}_i \tag{20}$$

$$\mathbf{w}_j^{\text{final}} = \mathbf{w}_j^{\text{recon}} + \Delta\mathbf{w}_j \tag{21}$$

By construction, the final restored weights, $\mathbf{w}_i^{\text{final}}$ and $\mathbf{w}_j^{\text{final}}$, are now guaranteed to be bit-for-bit identical to the original weights, thus ensuring the entire encoding-decoding pipeline is perfectly and demonstrably reversible.

### B.3 The Reversibility-Compression Trade-off

This residual correction method provides a definitive guarantee of lossless performance. However, it introduces an explicit trade-off between perfect reversibility and storage efficiency. The residual vectors have the same dimensionality as the neuron weights themselves, and storing them necessarily increases the size of the recovery instructions. This may run counter to the goal of achieving the highest possible compression ratio. The decision to implement this correction is therefore application-dependent: for contexts where absolute, provable fidelity is paramount, storing the residual is the correct and necessary approach. For applications where near-lossless reconstruction is sufficient, this step may be omitted to maximize compression.

### B.4 Ensuring Encoding Integrity: Cheating Detection via Residual Analysis

Storing the residual, however, creates a potential loophole. A faulty or poorly-converged GID factorization could fail to produce a meaningful structured representation. Instead, it might output trivial primitives (e.g., all zeros) and offload the entire signal into the residual term, $\Delta\mathbf{w}$. While this still permits perfect reconstruction, it defeats the purpose of the method, as the residual would carry the primary signal rather than a minor correction.

To detect this, a safeguard can be implemented. After each factorization, the encoder calculates the relative norm of the residual for each neuron in the pair: $\frac{\|\Delta\mathbf{w}\|}{\|\mathbf{w}_{\text{original}}\|}$. If this ratio exceeds a small tolerance threshold (e.g., $10^{-6}$), it signals that the residual contains a substantial portion of the original signal's energy. Although this scenario is unlikely with a correctly computed factorization, this check is a crucial safeguard against implementation errors or convergence failures.

# C ALGORITHMS

## C.1 SCA

**Algorithm 1** Analytical SCA for 2D Subspaces

**Require:**
Two linearly independent vectors $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^{n+c}$ spanning the subspace $S$.
Numerical tolerance $\varepsilon_{\text{angle}}$ (e.g., $10^{-8}$).
**Ensure:** The optimal SCA basis $\{\mathbf{p}^*, \mathbf{q}^*\}$ for $S$, where $\mathbf{p}^*$ is n-dominated and $\mathbf{q}^*$ is c-dominated.

**procedure** ANALYTICALSCA2D($\mathbf{w}_1, \mathbf{w}_2, n$)
▷ **Step 1: Construct an initial orthonormal basis $\{\mathbf{c}_1, \mathbf{c}_2\}$ for $S$ via Gram-Schmidt.**
$\mathbf{c}_1 \leftarrow \mathbf{w}_1 / \|\mathbf{w}_1\|$
$\mathbf{u}_2 \leftarrow \mathbf{w}_2 - (\mathbf{w}_2 \cdot \mathbf{c}_1)\mathbf{c}_1$
**if** $\|\mathbf{u}_2\| / \|\mathbf{w}_2\| < \varepsilon_{\text{angle}}$ **then**
**Print** "Warning: Input vectors are nearly parallel. Results may be inaccurate due to Gram-Schmidt instability."
**end if**
$\mathbf{c}_2 \leftarrow \mathbf{u}_2 / \|\mathbf{u}_2\|$
▷ **Step 2: Form the $2 \times 2$ Gram matrix $M$ from the n-space components.**
$(\mathbf{c}_1)_n \leftarrow$ first $n$ components of $\mathbf{c}_1$
$(\mathbf{c}_2)_n \leftarrow$ first $n$ components of $\mathbf{c}_2$
$a \leftarrow \|(\mathbf{c}_1)_n\|^2; \quad b \leftarrow (\mathbf{c}_1)_n \cdot (\mathbf{c}_2)_n; \quad c \leftarrow \|(\mathbf{c}_2)_n\|^2$
▷ **Step 3: Solve the eigenvalue problem for $M$ analytically to find orthonormal eigenvectors.**
$(\_, \mathbf{v}_1) \leftarrow$ SOLVE2X2EIGENPROBLEM($a, b, c$)
$\mathbf{v}_2 \leftarrow \begin{pmatrix} -(\mathbf{v}_1)_2 \\ (\mathbf{v}_1)_1 \end{pmatrix}$ ▷ Orthogonal complement of $\mathbf{v}_1$
▷ **Step 4: Construct the optimal SCA basis using the eigenvectors.**
$\mathbf{p}^* \leftarrow (\mathbf{v}_1)_1 \mathbf{c}_1 + (\mathbf{v}_1)_2 \mathbf{c}_2$ ▷ $\mathbf{p}^*$ is n-dominated as it's built from $\mathbf{v}v_1$.
$\mathbf{q}^* \leftarrow (\mathbf{v}_2)_1 \mathbf{c}_1 + (\mathbf{v}_2)_2 \mathbf{c}_2$ ▷ $\mathbf{q}^*$ is c-dominated as it's built from $\mathbf{v}v_2$.
**return** $\{\mathbf{p}^*, \mathbf{q}^*\}$
**end procedure**

### C.2 GID

**Algorithm 2** Compute GID Factorization

**Require:** Joint weight matrix $\mathbf{W}_{\text{joint}} \in \mathbb{R}^{(n+c)\times 2}$, dimension $n$.
**Ensure:** GID primitives $\{\mathbf{u}, \mathbf{v}, \mathbf{z}, \mathbf{r}\}$ and coefficients $\{\alpha_i, \beta_i, \alpha_j, \beta_j\}$, or `null` on failure..

**procedure** COMPUTEGID($\mathbf{W}_{\text{joint}}, n$)

▷ **Step 1: Dimensionality Reduction**

$\mathbf{W}_{in} \leftarrow \mathbf{W}_{\text{joint}}[1:n, :]$, $\mathbf{W}_{out} \leftarrow \mathbf{W}_{\text{joint}}[n+1:\text{end}, :]$
$\mathbf{Q}_n, \mathbf{R}_n \leftarrow$ QR_decomposition($\mathbf{W}_{in}$)
$\mathbf{Q}_c, \mathbf{R}_c \leftarrow$ QR_decomposition($\mathbf{W}_{out}$)
**if** $|\det(\mathbf{R}_c)| < \epsilon$ **then return** `null` ▷ Precondition: $\mathbf{W}_{out}$ must be full rank.
**end if**

▷ **Step 2: Form 2D Problem Matrices**

$\mathbf{P}^T \leftarrow$ SOLVE2X2LINEARSYSTEM($\mathbf{R}_c^T, \mathbf{R}_n^T$)
**if** $\mathbf{P}$ is null **then return** `null`
**end if**
$\mathbf{P}_{inv} \leftarrow$ SOLVE2X2LINEARSYSTEM($\mathbf{P}, \mathbf{I}$)
**if** $\mathbf{P}_{inv}$ is null **then return** `null`
**end if**
$\mathbf{A} \leftarrow \mathbf{P} - \mathbf{P}_{inv}^T$
$\mathbf{M} \leftarrow \mathbf{A}^T\mathbf{J}^T\mathbf{P}$, where $\mathbf{J} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$
$\mathbf{C} \leftarrow \mathbf{A}^T\mathbf{A}$, $\mathbf{M}_{sym} \leftarrow 0.5(\mathbf{M} + \mathbf{M}^T)$

▷ **Step 3: Find Optimal Direction**

$\mathbf{p} \leftarrow$ GETQUARTICCOEFFICIENTS($\mathbf{C}, \mathbf{M}_{sym}$)
$\hat{\mathbf{u}}^* \leftarrow$ FINDOPTIMALDIRECTION($\mathbf{p}, \mathbf{C}, \mathbf{M}_{sym}$)
**if** $\hat{\mathbf{u}}^*$ is null **then return** `null`
**end if**

▷ **Step 4: Reconstruct 2D Primitives**

$N^* \leftarrow (\hat{\mathbf{u}}^*)^\top \mathbf{C}\hat{\mathbf{u}}^*$, $D^* \leftarrow (\hat{\mathbf{u}}^*)^\top \mathbf{M}_{sym}\hat{\mathbf{u}}^*$
$\rho^{*2} \leftarrow \sqrt{N^*}/|D^*|$
$\mathbf{r}_{2d}^* \leftarrow \sqrt{\rho^{*2}} \cdot \hat{\mathbf{u}}^*$
$k^* \leftarrow 1/(\rho^{*2} D^*)$
$\mathbf{z}_{2d}^* \leftarrow k^* \mathbf{J}\mathbf{A}\mathbf{r}_{2d}^*$
$\mathbf{u}_{2d}^{*,\text{unnorm}} \leftarrow \mathbf{P}\mathbf{r}_{2d}^*$
$\mathbf{v}_{2d}^{*,\text{unnorm}} \leftarrow$ SOLVE2X2LINEARSYSTEM($\mathbf{P}, \mathbf{z}_{2d}^*$)
**if** $\mathbf{v}_{2d}^{*,\text{unnorm}}$ is null **then return** `null`
**end if**
$\mathbf{u}_{2d}^* \leftarrow \mathbf{u}_{2d}^{*,\text{unnorm}} / \left\|\mathbf{u}_{2d}^{*,\text{unnorm}}\right\|$, $\mathbf{v}_{2d}^* \leftarrow \mathbf{v}_{2d}^{*,\text{unnorm}} / \left\|\mathbf{v}_{2d}^{*,\text{unnorm}}\right\|$

▷ **Step 5: Reconstruct Full-Dimensional Primitives**

$\{\mathbf{u}, \mathbf{v}, \mathbf{z}, \mathbf{r}\} \leftarrow \{\mathbf{Q}_n\mathbf{u}_{2d}^*, \mathbf{Q}_c\mathbf{v}_{2d}^*, \mathbf{Q}_n\mathbf{z}_{2d}^*, \mathbf{Q}_c\mathbf{r}_{2d}^*\}$

▷ **Step 6: Solve for Coefficients Analytically**

Construct GID basis $\mathbf{B} = [\mathbf{p}_{gid}|\mathbf{q}_{gid}]$.
$\boldsymbol{y}_i \leftarrow \mathbf{B}^T\mathbf{w}_i$, $\boldsymbol{y}_j \leftarrow \mathbf{B}^T\mathbf{w}_j$
$\mathbf{G} \leftarrow \mathbf{B}^T\mathbf{B}$
**if** $|\det(\mathbf{G})| < \epsilon$ **then return** `null`
**end if**
$(\alpha_i, \beta_i)^\top \leftarrow$ SOLVE2X2LINEARSYSTEM($\mathbf{G}, \boldsymbol{y}_i$)
$(\alpha_j, \beta_j)^\top \leftarrow$ SOLVE2X2LINEARSYSTEM($\mathbf{G}, \boldsymbol{y}_j$)
**if** $(\alpha_i, \beta_i)$ is null **or** $(\alpha_j, \beta_j)$ is null **then return** `null`
**end if**

▷ **Step 7: Final Sanity Check**

**if** any primitive or coefficient contains NaN or Inf **then return** `null`
**end if**
**return** GID primitives $\{\mathbf{u}, \mathbf{v}, \mathbf{z}, \mathbf{r}\}$ and coefficients $\{\alpha_i, \beta_i, \alpha_j, \beta_j\}$.
**end procedure**

### C.3 HIERARCHICAL CODING

**Algorithm 3** Reconstruct Neuron from GID Primitives

**Require:** GID primitives $\{\mathbf{u}, \mathbf{v}, \mathbf{z}, \mathbf{r}\}$, coefficients $\alpha, \beta$.
**Ensure:** Reconstructed joint weight vector $\mathbf{w}^{\text{recon}}$.
**procedure** RECONSTRUCTFROMGID($\{\mathbf{u}, \mathbf{v}, \mathbf{z}, \mathbf{r}\}, \alpha, \beta$)
  $\mathbf{w}_n \leftarrow \alpha\mathbf{u} + \beta\langle\mathbf{v}, \mathbf{r}\rangle\mathbf{z}$
  $\mathbf{w}_c \leftarrow \beta\mathbf{v} + \alpha\langle\mathbf{u}, \mathbf{z}\rangle\mathbf{r}$
  **return** $\begin{pmatrix}\mathbf{w}_n \\ \mathbf{w}_c\end{pmatrix}$
**end procedure**

The encoder iteratively simplifies a given layer by applying the following four-step cycle until only one neuron remains:

1. **Select Pair:** For the current layer, find the optimal neuron pair $(i^*, j^*)$ to merge. This is the pair that minimizes symmetric structural proxy $\frac{1}{2}\left(C_{\text{struct}}(i, j) + C_{\text{struct}}(j, i)\right)$ among all pairs that are deemed *eligible* (i.e. satisfying the three conditions on its Group Normalization $\gamma$ parameters defined in Section A.4).
2. **Direction Assignment:** Determine the optimal merge direction by finding the minimum of $C_{\text{struct}}(i^*, j^*)$ and $C_{\text{struct}}(j^*, i^*)$. Compute $\mathbf{w}^*_{\text{new}}$ for this optimal assignment.
3. **Store Instructions:** Archive a recovery instruction $I_{i^*j^*}$ containing all information required for a perfect reversal of the merge:
   - The location of the merged pair (including the layer index): $(i, j, \ell)$.
   - The GID primitives: $\{\mathbf{u}, \mathbf{v}, \mathbf{z}, \mathbf{r}\}$.
   - The reconstruction coefficients: $\{\alpha_{i^*}, \beta_{i^*}, \alpha_{j^*}, \beta_{j^*}\}$.
   - The original Group Normalization parameters: $\{\gamma_{i^*}, \delta_{i^*}, \gamma_{j^*}, \delta_{j^*}\}$.
   - (Optional) The numerical residuals, $\Delta\mathbf{w}_{i^*}$ and $\Delta\mathbf{w}_{j^*}$, to ensure bit-perfect reversibility.
4. **Network Update:** Simplify the network by setting the designated victim neuron's weights to $\mathbf{0}$ and replacing the survivor neuron's weights with the computed $\mathbf{w}^*_{\text{new}}$. The *effective* weights of this new neuron are constructed from the primary GID components:

$$(\mathbf{w}^*_{\text{new}})^{\text{in}}_{\text{eff}} \triangleq \alpha^*_0\mathbf{u} \quad \text{and} \quad (\mathbf{w}^*_{\text{new}})^{\text{out}}_{\text{eff}} \triangleq \beta^*_0\mathbf{v}$$

   where $\alpha^*_0, \beta^*_0$ are determined by (13). The final *bare* kernel weights are then computed by reversing the Group Normalization scaling with the new parent gamma, $\gamma_{\text{new}} = \frac{1}{2}(\gamma_{i^*} + \gamma_{j^*})$.

### C.4 THE DECODER: PERFECT RECONSTRUCTION VIA ALGEBRAIC REVERSAL

The decoder's function is to reverse the encoding process, perfectly reconstructing the original, dense neural network from its compressed skeletal structure and the ordered sequence of recovery instructions. This is achieved by applying the instructions in the reverse order of their creation, systematically executing an "un-merging" operation at each step until the network is restored to its full, pre-trained state. Each un-merging step is a deterministic algebraic substitution that perfectly restores two "child" neurons from a single "parent" neuron, demonstrating that the encoding is fully reversible.

**Proposition 1 (Perfect Reversibility of the Un-Merge Operation)** *The un-merge operation is the exact algebraic inverse of the GID factorization performed during encoding. Given an instruction set $I_{ij}$, the decoder can perfectly reconstruct the original neuron pair $(\mathbf{w}_i, \mathbf{w}_j)$.*

**Proof** Let the encoder map $(\mathbf{w}_i, \mathbf{w}_j) \to (\mathbf{w}_{\text{new}}, I_{ij})$. The decoder receives the instruction packet $I_{ij}$ and the current state of the simplified network. The reconstruction for each child neuron $h \in \{i, j\}$ proceeds in four deterministic steps, using only the information in $I_{ij}$:

1. **Direct Recovery of Primitives:** The decoder retrieves the *complete* set of original GID primitives $\{\mathbf{u}, \mathbf{v}, \mathbf{z}, \mathbf{r}\}$ and reconstruction coefficients $\{\alpha_h, \beta_h\}$ directly from the instruction packet $I_{ij}$. This direct retrieval protocol is robust and avoids the numerical instability of inferring the primary vectors from the parent neuron's weights.
2. **Algebraic Reconstruction of Effective Weights:** Using the recovered primitives, the decoder algebraically reconstructs the *effective* weights of the child neuron, $(\mathbf{w}_h)_{\text{eff}}$, which represent the weights after Group Normalization scaling has been applied.

$$(\mathbf{w}_h)^{\text{in, rec}}_{\text{eff}} = \alpha_h\mathbf{u} + \beta_h\langle\mathbf{v}, \mathbf{r}\rangle\mathbf{z} = (\mathbf{w}_h)^{\text{in}}_{\text{eff}}$$

$$(\mathbf{w}_h)^{\text{out, rec}}_{\text{eff}} = \beta_h\mathbf{v} + \alpha_h\langle\mathbf{u}, \mathbf{z}\rangle\mathbf{r} = (\mathbf{w}_h)^{\text{out}}_{\text{eff}}$$

3. **Reversal of Group Normalization:** To obtain the final *bare* kernel weights that are stored in the model, the decoder must reverse the effect of the Group Normalization layer.[1] It retrieves the original scalar GN parameter $\gamma_h$ from the instruction packet and performs the inverse scaling operation:

$$(\mathbf{w}_h)^{\text{bare}}_n = \frac{(\mathbf{w}_h)^{\text{in, rec}}_{\text{eff}}}{\gamma_h}$$

   The c-space weights, $(\mathbf{w}_h)_c$, are already bare and require no modification.

[1] The other learnable parameter of Group Normalization, the additive shift $\delta_h$, is not part of this multiplicative reversal. It is treated as a simple bias term that is not included in the GID factorization. Instead, its original value is archived in the instruction packet by the encoder and restored directly by the decoder, separate from this weight reconstruction process.

**Algorithm 4** Hierarchical Encoder

**Input:** Initial weights $W$, Paired layer definitions $\mathcal{P}$, Active neuron indices $N$
**Output:** Skeletal weights $W_{\text{skel}}$, Instruction queue $\mathcal{I}$
$\mathcal{I} \leftarrow [\,]$ ▷ LIFO queue for instructions
$E \leftarrow \{\text{empty dictionary of sets}\}$ ▷ Exclusion list for failed pairs
**while** true **do**
  $best\_merge \leftarrow \{\text{cost: } \infty\}$
  ▷ **– Step 1: Find Global Best Pair (SMC) –**
  **for** $p \in \mathcal{P}$ **do** ▷ Global "free-for-all" search
    $L_m \leftarrow \text{NAME}(p)$
    **if** $|N_{L_m}| \leq 1$ **then**
      **continue**
    **end if**
    $W_{\text{cost}} \leftarrow \text{GETJOINTWEIGHTSFORCOST}(W, p)$
    **for** $(i, j) \in \text{PAIRS}(N_{L_m})$ **do**
      **if** $(i, j) \in E_{L_m}$ **then**
        **continue**
      **end if** ▷ Skip failed pairs
      $c_{ij} \leftarrow \text{CSTRUCT}(W_{\text{cost}}[i], W_{\text{cost}}[j])$
      $c_{ji} \leftarrow \text{CSTRUCT}(W_{\text{cost}}[j], W_{\text{cost}}[i])$
      $s \leftarrow \min(c_{ij}, c_{ji})$ ▷ **SMC scoring metric**
      **if** $s < best\_merge$.cost **then**
        $best\_merge \leftarrow \{\text{cost: } s, \text{pair: } (i, j), \text{layer: } p, \text{costs: } (c_{ij}, c_{ji})\}$
      **end if**
    **end for**
  **end for**
  **if** $best\_merge$.cost $= \infty$ **then**
    **break**
  **end if** ▷ No valid merges left
  ▷ **– Step 2: GID Factorization & Eligibility Check –**
  $(i, j) \leftarrow best\_merge$.pair, $p^* \leftarrow best\_merge$.layer
  $L_{\text{current}} \leftarrow \text{NAME}(p^*)$
  **if** $best\_merge$.costs[0] $\leq best\_merge$.costs[1] **then**
    $(i^*, j^*) \leftarrow (i, j)$ ▷ i is survivor, j is victim
  **else**
    $(i^*, j^*) \leftarrow (j, i)$ ▷ j is survivor, i is victim
  **end if**
  $W_{\text{merge}} \leftarrow \text{GETJOINTWEIGHTS}(W, p^*)$ ▷ Use asymmetric weights for merge
  $W_{\text{pair}} \leftarrow [W_{\text{merge}}[i^*], W_{\text{merge}}[j^*]]$
  $gid\_result \leftarrow \text{COMPUTEGID}(W_{\text{pair}})$
  $eligible \leftarrow \text{CHECKELIGIBILITY}(W, i^*, j^*, \text{STRATEGY})$
  **if** $gid\_result = \text{null}$ **or not** $eligible$ **then**
    $E_{L_{\text{current}}}$.add(sorted$(i, j)$)
    **continue** ▷ Restart search
  **end if**
  ▷ **– Step 3: Archive Instructions and Merge –**
  $(\alpha_0, \beta_0) \leftarrow \text{COMPUTEMERGECOEFFS}(W_{\text{pair}}, gid\_result, \text{STRATEGY})$
  $(\gamma_{\text{new}}, \delta_{\text{new}}) \leftarrow \text{COMPUTEGNPARAMS}(W, (i^*, j^*), \text{STRATEGY})$
  $\mathbf{w}_{\text{new}} \leftarrow \text{CONSTRUCTMERGEDNEURON}(\alpha_0, \beta_0, gid\_result)$
  $(\Delta\mathbf{w}_i, \Delta\mathbf{w}_j) \leftarrow \text{COMPUTERESIDUALS}(W_{\text{pair}}, gid\_result)$
  $I \leftarrow \text{CREATEPACKET}((i^*, j^*), gid\_result, \text{coeffs}, \Delta\mathbf{w}, \dots)$
  $\mathcal{I}$.push$(I)$
  $W \leftarrow \text{UPDATENETWORKSTATE}(W, (i^*, j^*), \mathbf{w}_{\text{new}}, (\gamma_{\text{new}}, \delta_{\text{new}}))$
  $N_{L_{\text{current}}}$.remove$(j^*)$ ▷ Remove victim
**end while**
$\mathcal{I}$.reverse()
**return** $W, \mathcal{I}$

4. **Residual Correction for Numerical Precision:** Finally, the decoder adds the stored numerical residual vector $\Delta\mathbf{w}_h$ from the instruction packet back to the reconstructed weights.

$$\mathbf{w}_h^{\text{final}} = \mathbf{w}_h^{\text{rec}} + \Delta\mathbf{w}_h$$

This step corrects for any minuscule floating-point errors from the encoder's GID factorization, ensuring the final result is bit-for-bit identical to the original weight vector $\mathbf{w}_h$.

After computing the final child neuron weights, the decoder updates the simplified network by replacing the parent neuron with the two reconstructed child neurons. Since all primitives are retrieved directly and all architectural transformations are explicitly reversed, the un-merge operation is a perfect reversal.

□

**Algorithm 5** Hierarchical Decoder

**Input:** A skeletal layer and a list of recovery instructions $\mathcal{I}$.
**Output:** The original, fully reconstructed network layer.
**for** each instruction $I$ in $\mathcal{I}$ (in order) **do**
▷ **– Step 1: Recover All Original Parameters from Instruction –**
  $(i, j, \mathbf{u}, \mathbf{v}, \mathbf{z}, \mathbf{r}, \alpha_i, \beta_i, \alpha_j, \beta_j, \gamma_i, \delta_i, \gamma_j, \delta_j, \Delta\mathbf{w}_i, \Delta\mathbf{w}_j) \leftarrow I$.
  ▷ **– Step 2: Reconstruct Child Neurons –**
  **for** each child neuron $h \in \{i, j\}$ **do**
    ▷ First, reconstruct the effective weights from GID primitives.
    $(\mathbf{w}_h^{\text{recon}})_n \leftarrow \alpha_h\mathbf{u} + \beta_h\langle\mathbf{v}, \mathbf{r}\rangle\mathbf{z}$
    $(\mathbf{w}_h^{\text{recon}})_c \leftarrow \beta_h\mathbf{v} + \alpha_h\langle\mathbf{u}, \mathbf{z}\rangle\mathbf{r}$
    ▷ Second, apply the stored residual to achieve bit-perfect effective weights.
    $\mathbf{w}_h^{\text{eff}} \leftarrow \mathbf{w}_h^{\text{recon}} + \Delta\mathbf{w}_h$
    ▷ Finally, reverse the Group Normalization to get the final bare weights.
    $(\mathbf{w}_h^{\text{final}})_n \leftarrow (\mathbf{w}_h^{\text{eff}})_n / \gamma_h$
    $(\mathbf{w}_h^{\text{final}})_c \leftarrow (\mathbf{w}_h^{\text{eff}})_c$ ▷ c-space weights are already bare.
  **end for**
  ▷ **– Step 3: Update the Network State –**
  Expand the network layer to accommodate two neurons at locations $i$ and $j$.
  Restore n-space weights of neurons $i, j$ with $(\mathbf{w}_i^{\text{final}})_n, (\mathbf{w}_j^{\text{final}})_n$.
  Restore c-space weights for neurons $i, j$ with $(\mathbf{w}_i^{\text{final}})_c, (\mathbf{w}_j^{\text{final}})_c$.
  Restore bias params for neurons $i, j$ with $(b_i, b_j)$.
  Restore GN params for neurons $i, j$ with $(\gamma_i, \delta_i), (\gamma_j, \delta_j)$.
**end for**
**return** The fully reconstructed layer.

### C.5 DEFT

**Algorithm 6** Generate Elasticity Map

**Require:** HERM decoding curriculum $\mathcal{I} = \{I_1, \ldots, I_M\}$, expert model $f_S$.
**Ensure:** Elasticity map $\mathcal{C}$.
Initialize $\mathcal{C}$ as a map from parameter paths to zero-tensors with shapes matching $f_S$.
$M \leftarrow$ length($\mathcal{I}$).
**for** $m \in \{1, \ldots, M\}$ **do**
  $e_m \leftarrow m/M$.
  Extract GID primitives $\{\mathbf{u}, \mathbf{v}, \mathbf{z}, \mathbf{r}\}$ and coeffs from $I_m$.
  Reconstruct interactional components (e.g., $\mathbf{w}_{i,\text{int}}^n \leftarrow \beta_i \langle\mathbf{v}, \mathbf{r}\rangle \mathbf{z}$).
  **for** each interactional component vector $\mathbf{w}_{\text{int}}$ **do**
    $\mathbf{w}_{\text{mask}} \leftarrow |\mathbf{w}_{\text{int}}| / (\|\mathbf{w}_{\text{int}}\|_2 + \epsilon)$.
    Reshape $\mathbf{w}_{\text{mask}}$ to match its corresponding kernel tensor slice.
    Update the relevant slice of $\mathcal{C}$[kernel_path] via $\max(\text{current\_value}, \mathbf{w}_{\text{mask}} \cdot e_m)$.
  **end for**
**end for**
▷ Post-processing: Propagate elasticity to other trainable parameters.
**for** each layer in $f_S$ **do**
  **if** layer has a kernel **then**
    Compute per-neuron max elasticity from $\mathcal{C}$[kernel_path].
    Assign this max value to corresponding slices in $\mathcal{C}$[bias_path], $\mathcal{C}$[beta_path], and $\mathcal{C}$[gamma_path].
  **end if**
**end for**
**return** $\mathcal{C}$.

#### C.5.1 FUNCTION 2: TRAIN_STEP_DEFT

This function performs a single step of fine-tuning using the generated map.

### C.6 SELF-ORGANIZING NETWORK TRAINING

## D MATHEMATICAL RESULTS

**Proposition 2 (Structural Component Analysis)**

**Algorithm 7** DEFT Training Step

**Require:** Current parameters $\boldsymbol{\theta}$, data batch $(x, y)$, elasticity map $\mathcal{C}$.
**Ensure:** Updated parameters $\boldsymbol{\theta}'$.
1: Compute gradients: $\nabla L_D \leftarrow \nabla_{\boldsymbol{\theta}} L_D(\boldsymbol{\theta}; x, y)$.
2: Modulate gradients: $\nabla L_{\text{DEFT}} \leftarrow \nabla L_D \odot \mathcal{C}$ (element-wise product).
3: Apply updates: $\boldsymbol{\theta}' \leftarrow \text{OptimizerUpdate}(\boldsymbol{\theta}, \nabla L_{\text{DEFT}})$.
4: **return** $\boldsymbol{\theta}'$.

**Algorithm 8** Self-Organizing Network Training Loop

1: Initialize a small skeletal network $\mathcal{M}$.
2: **for** each training phase $t = 1, \ldots, T$ **do**
3: Train $\mathcal{M}$ for $N$ epochs using standard gradient descent.
4: **Assess State:**
5: Compute loss trajectory $\mathcal{L}_{val}^{(t)}$.
6: Compute network activity statistics (e.g., mean activations, gradient norms).
7: **if** loss has plateaued for $P$ epochs **then**
8: **Expand:** Select neuron $k^*$ to split via $\text{argmax}_k \|\nabla_{\mathbf{w}_k} L\|$.
9: Replace $k^*$ with a new pair $(i, j)$ via the GID-based splitting mechanism.
10: **else if** network exhibits signs of over-capacity **then**
11: **Shrink:** Select pair $(i^*, j^*)$ to merge via $\text{argmin}_{i,j} C_{\text{true}}(i, j)$.
12: Replace $(i^*, j^*)$ with a single neuron $\mathbf{w}_{\text{new}}$ via the HERM merge operation.
13: **end if**
14: **end for**

***Setup.*** *Given a k-dimensional subspace $S \subseteq \mathbb{R}^{n+c}$, where $n, c > 0$ and $k \geq 2$. We view the space as the product $\mathbb{R}^n \times \mathbb{R}^c$, associating the first $n$ components of any vector with $\mathbb{R}^n$ and the remaining $c$ components with $\mathbb{R}^c$. Let $k_n$ be an integer hyperparameter such that $1 \leq k_n < k$. We seek to find an orthonormal basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$ for $S$ that minimizes the structural coupling cost function:*

$$J(\mathbf{b}_1, \ldots, \mathbf{b}_k) = \sum_{i=1}^{k_n} \|(\mathbf{b}_i)_c\|^2 + \sum_{j=k_n+1}^{k} \|(\mathbf{b}_j)_n\|^2$$

*where $(\mathbf{b}_j)_n \in \mathbb{R}^n$ and $(\mathbf{b}_j)_c \in \mathbb{R}^c$ are the components of the basis vector $\mathbf{b}_j \in \mathbb{R}^{n+c}$. The first $k_n$ vectors are designated to be n-dominated, and the remaining $k - k_n$ vectors are designated to be c-dominated.* ***Existence and Uniqueness.*** *An optimal basis that minimizes this cost function is always guaranteed to exist. The uniqueness of the solution depends on the eigenspectrum of the matrix $M$ defined below:*

1. ***Simple Eigenvalues:*** *If all eigenvalues of the matrix $M$ are distinct, the set of optimal basis vectors $\{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$ is unique up to the $2^k$ possible combinations of flipping the sign of each vector.*

2. ***Degenerate Eigenvalues:*** *If an eigenvalue of $M$ is repeated with multiplicity $d > 1$, the corresponding $d$ optimal basis vectors are not unique. They possess an additional continuous degree of freedom corresponding to any arbitrary rotation within the $d$-dimensional eigenspace, an $SO(d)$ rotational freedom.*

***Analytical Form of the Solution.*** *An optimal basis can be constructed as follows:*

1. ***Construct an Initial Basis:*** *Find an arbitrary orthonormal basis $\{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$ for the subspace $S$. Decompose each vector into its components: $\mathbf{e}_i = [(\mathbf{e}_i)_n^\top, (\mathbf{e}_i)_c^\top]^\top$.*

2. ***Form the Gram Matrix:*** *Construct the $k \times k$ symmetric matrix $M$ from the n-space components of the initial basis, where the elements of $M$ are given by the dot products:*

$$M_{ij} = (\mathbf{e}_i)_n \cdot (\mathbf{e}_j)_n$$

3. ***Solve the Eigenvalue Problem:*** *Find the eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ and their corresponding real eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k$ for the matrix $M$. These eigenvectors are themselves orthonormal vectors in $\mathbb{R}^k$.*

4. ***Construct the Optimal Basis:*** *The optimal basis vectors $\{\mathbf{b}_1^*, \ldots, \mathbf{b}_k^*\}$ are linear combinations of the initial basis vectors, with coefficients given by the components of the eigenvectors of $M$:*

$$\mathbf{b}_j^* = \sum_{i=1}^{k} (\mathbf{v}_j)_i \mathbf{e}_i$$

*The first $k_n$ vectors of this new basis, $\{\mathbf{b}_1^*, \ldots, \mathbf{b}_{k_n}^*\}$, corresponding to the $k_n$ largest eigenvalues of $M$, are the optimal n-dominated vectors. The remaining $k - k_n$ vectors, $\{\mathbf{b}_{k_n+1}^*, \ldots, \mathbf{b}_k^*\}$, are the optimal c-dominated vectors.*

**Proof** Let $\{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$ be an arbitrary orthonormal basis for $S$. Any other orthonormal basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$ for $S$ can be expressed as a rotation of the initial basis, defined by a $k \times k$ orthogonal matrix $R$, such that $\mathbf{b}_j = \sum_{i=1}^{k} R_{ij} \mathbf{e}_i$. Let $\mathbf{r}_j$ be the j-th column of $R$. The set $\{\mathbf{r}_1, \ldots, \mathbf{r}_k\}$ forms an orthonormal basis for $\mathbb{R}^k$. The squared norm of the n-space component of a new basis vector is given by the quadratic form $\|(\mathbf{b}_j)_n\|^2 = \mathbf{r}_j^\top M \mathbf{r}_j$, where $M$ is the Gram matrix $M_{ij} = (\mathbf{e}_i)_n \cdot (\mathbf{e}_j)_n$. Since $\mathbf{b}_j$ is a unit vector, we have $\|(\mathbf{b}_j)_c\|^2 = 1 - \|(\mathbf{b}_j)_n\|^2$. Substituting this into the

cost function $J$ yields:

$$J = \sum_{i=1}^{k_n}(1 - \|(\mathbf{b}_i)_n\|^2) + \sum_{j=k_n+1}^{k} \|(\mathbf{b}_j)_n\|^2$$
$$= k_n - \left(\sum_{i=1}^{k_n} \|(\mathbf{b}_i)_n\|^2 - \sum_{j=k_n+1}^{k} \|(\mathbf{b}_j)_n\|^2\right)$$

Minimizing $J$ is equivalent to maximizing the quantity $J'$ as defined below.

$$J' \triangleq \sum_{i=1}^{k_n} \|(\mathbf{b}_i)_n\|^2 - \sum_{j=k_n+1}^{k} \|(\mathbf{b}_j)_n\|^2 . \tag{22}$$

The objective function $J'$ can be rewritten using the trace operator. Since $\boldsymbol{r}^\top M \boldsymbol{r}$ is a scalar, it equals its own trace. By the cyclic property of the trace, $\mathrm{Tr}(\boldsymbol{r}^\top M \boldsymbol{r}) = \mathrm{Tr}(M \boldsymbol{r}\boldsymbol{r}^\top)$. The objective function can therefore be expressed as:

$$J' = \mathrm{Tr}\left(M\left(\sum_{i=1}^{k_n} \boldsymbol{r}_i \boldsymbol{r}_i^\top - \sum_{j=k_n+1}^{k} \boldsymbol{r}_j \boldsymbol{r}_j^\top\right)\right)$$

Let's define the matrix $Q \triangleq \sum_{i=1}^{k_n} \boldsymbol{r}_i \boldsymbol{r}_i^\top - \sum_{j=k_n+1}^{k} \boldsymbol{r}_j \boldsymbol{r}_j^\top$. We seek to choose the orthonormal basis $\{\boldsymbol{r}_j\}$ to maximize $\mathrm{Tr}(MQ)$.

The basis vectors $\{\boldsymbol{r}_j\}$ are the eigenvectors of $Q$. Specifically, for any $l \le k_n$, $Q\boldsymbol{r}_l = \boldsymbol{r}_l$, giving an eigenvalue of $+1$. For any $l > k_n$, $Q\boldsymbol{r}_l = -\boldsymbol{r}_l$, giving an eigenvalue of $-1$. By the Von Neumann trace inequality, the expression $\mathrm{tr}(MQ)$ is maximized when the eigenspaces of the symmetric matrices $M$ and $Q$ are aligned according to their eigenvalue ordering. Let $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k\}$ be the orthonormal eigenvectors of $M$ with corresponding real eigenvalues sorted as $\lambda_1 \ge \cdots \ge \lambda_k$. To maximize the trace, the eigenvectors of $Q$ with the largest eigenvalue $(+1)$ must be chosen to be the eigenvectors of $M$ with the largest eigenvalues. Therefore, the optimal choice for maximizing $J' = \mathrm{tr}(MQ)$ is to set the basis $\{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{k_n}\}$ to be the eigenvectors $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{k_n}\}$ of $M$. This makes $\|(\boldsymbol{b}_i)_n\|^2 = \boldsymbol{v}_i^\top M \boldsymbol{v}_i = \lambda_i$ for $i = 1, \ldots, k_n$. The remaining basis vectors $\{\boldsymbol{r}_{k_n+1}, \ldots, \boldsymbol{r}_k\}$ are consequently the remaining eigenvectors of $M$, making $\|(\boldsymbol{b}_j)_n\|^2 = \lambda_j$ for $j > k_n$. This choice of rotation matrix $R$ (whose columns are the eigenvectors of $M$ in order) constructs the optimal basis $\{\mathbf{b}_j\}$ as claimed.

**Proof of Uniqueness.** The uniqueness of the basis $\{\mathbf{b}_j\}$ is determined by the uniqueness of the eigenvectors $\{\mathbf{v}_j\}$ of the real symmetric matrix $M$.

1. **Case of Simple Eigenvalues:** If all eigenvalues $\lambda_1 > \lambda_2 > \cdots > \lambda_k$ are distinct, then the corresponding eigenspace for each $\lambda_j$ is one-dimensional. This means the direction of each eigenvector $\mathbf{v}_j$ is uniquely determined. The constraint that $\|\mathbf{v}_j\| = 1$ restricts the choice to two vectors, $\mathbf{v}_j$ and $-\mathbf{v}_j$. Since the optimal basis vectors $\mathbf{b}_j$ are constructed from these eigenvectors, the ambiguity is limited to a sign flip for each $\mathbf{b}_j$.
2. **Case of Degenerate Eigenvalues:** If an eigenvalue $\lambda$ is repeated with multiplicity $d > 1$, its corresponding eigenspace $V_\lambda$ is $d$-dimensional. The solution requires choosing any $d$ orthonormal vectors that form a basis for $V_\lambda$. If $\{\mathbf{v}_1, \ldots, \mathbf{v}_d\}$ is one such basis, then any other basis $\{\mathbf{v}'_1, \ldots, \mathbf{v}'_d\}$ for $V_\lambda$ can be obtained by a rotation: $\mathbf{v}'_j = \sum_{l=1}^{d} O_{lj}\mathbf{v}_l$, where $O$ is a $d \times d$ orthogonal matrix. This continuous rotational freedom in the choice of eigenvectors translates directly to a continuous rotational freedom in the corresponding optimal basis vectors $\{\mathbf{b}_j\}$.

Thus, the solution is unique up to sign flips if and only if all eigenvalues of $M$ are simple. □

**Proposition 3 (Equivalence of Interaction Mode Definitions)** *A 2D subspace $S \subseteq \mathbb{R}^{n+c}$ has a **One-Sided Interaction** (as defined by its SCA basis) if and only if the dimensions of its projections are $\{\dim(S_n), \dim(S_c)\} = \{1, 2\}$.*

**Proof** Let $\{\mathbf{p}^*, \mathbf{q}^*\}$ be the orthonormal SCA basis for $S$.

($\Rightarrow$) Assume $S$ has a one-sided interaction. Without loss of generality, let $(\mathbf{q}^*)_n = \mathbf{0}$ and $(\mathbf{p}^*)_n \neq \mathbf{0}$. The projection of any vector $\mathbf{v} \in S$ onto the n-space is given by $\mathbf{v}_n = (c_1\mathbf{p}^* + c_2\mathbf{q}^*)_n = c_1(\mathbf{p}^*)_n + c_2(\mathbf{q}^*)_n = c_1(\mathbf{p}^*)_n$. Thus, the entire n-projection of the subspace is $S_n = \mathrm{span}((\mathbf{p}^*)_n)$, which is 1-dimensional. Since the basis vectors are linearly independent, the c-projection $S_c = \mathrm{span}((\mathbf{p}^*)_c, (\mathbf{q}^*)_c)$ must be 2-dimensional (otherwise the entire subspace would be 1D). Therefore, $\{\dim(S_n), \dim(S_c)\} = \{1, 2\}$.

($\Leftarrow$) Assume the projection dimensions are $\{1, 2\}$. Let $\dim(S_n) = 1$. This means the n-components of all vectors in $S$ are collinear, i.e., $S_n = \mathrm{span}(\mathbf{u})$ for some vector $\mathbf{u}$. We can therefore choose an orthonormal basis $\{\mathbf{p}, \mathbf{q}\}$ for $S$ such that $\mathbf{p}_n$ is aligned with $\mathbf{u}$ and $\mathbf{q}_n$ is orthogonal to it, which forces $\mathbf{q}_n = \mathbf{0}$. This basis has a one-sided structure. The SCA basis, being the one that minimizes coupling, must be at least this good, and will therefore also have a zero coupling component, satisfying the SCA definition of a one-sided interaction. □

**Proposition 4** *A GID basis can only represent a 2D subspace $S$ if the GID's structural mode (**Two-Sided**, **One-Sided**, or **No Interaction**) matches the intrinsic interaction mode of $S$.*

**Proof** The proof rests on a fundamental principle: for a GID basis $\{\mathbf{p}_g, \mathbf{q}_g\}$ to be a valid representation of a subspace $S$, the subspace it spans, $S_g$, must be identical to $S$. A necessary condition for this is that the dimensions of their respective projections must match, i.e., $\dim((S_g)_n) = \dim(S_n)$ and $\dim((S_g)_c) = \dim(S_c)$. We will prove by contradiction that this condition forces the GID's structural mode to match the intrinsic mode of $S$.

1. **Case 1: Subspace $S$ is Two-Sided** ($\dim(S_n) = 2, \dim(S_c) = 2$)**.**
   A valid GID representation for $S$ must generate 2-dimensional projections.

- *Contradiction for a One-Sided GID:* By definition, a one-sided GID has one zero coupling gate, for instance, $\langle \mathbf{v}, \mathbf{r} \rangle = 0$. This forces the GID basis vector $\mathbf{q}_g$ to have a zero n-component: $\mathbf{q}_n = \mathbf{0}$. The n-projection of the spanned subspace is then $(S_g)_n = \text{span}(\mathbf{p}_n, \mathbf{0}) = \text{span}(\mathbf{p}_n)$, which is at most 1-dimensional. This contradicts the requirement.
- *Contradiction for a No-Interaction GID:* By definition, both coupling gates are zero. This forces $\mathbf{q}_n = \mathbf{0}$ and $\mathbf{p}_c = \mathbf{0}$. The n-projection is therefore 1-dimensional, which is a contradiction.

Only a **Two-Sided GID**, where both gates are non-zero and the primitive vectors $\{\mathbf{u}, \mathbf{z}\}$ are linearly independent, can generate a 2-dimensional n-projection. Thus, it is the only valid representation.

2. **Case 2: Subspace $S$ is One-Sided (e.g., $\dim(S_n) = 1, \dim(S_c) = 2$).**
A valid GID representation must match these specific projection dimensions.

   - *Contradiction for a Two-Sided GID:* A two-sided GID, by requiring linearly independent primitives $\{\mathbf{u}, \mathbf{z}\}$, generates a 2-dimensional n-projection. This contradicts the requirement that $\dim(S_n) = 1$.
   - *Contradiction for a No-Interaction GID:* A no-interaction GID generates a 1-dimensional c-projection, as $(S_g)_c = \text{span}(\mathbf{0}, \mathbf{v}) = \text{span}(\mathbf{v})$. This contradicts the requirement that $\dim(S_c) = 2$.

   Therefore, only a **One-Sided GID** can produce the required ‘1, 2‘ projection dimensions.

3. **Case 3: Subspace $S$ has No Interaction ($\dim(S_n) = 1, \dim(S_c) = 1$).**
A valid GID must generate 1-dimensional projections in both spaces.

   - *Contradiction for a Two-Sided GID:* Generates 2-dimensional projections.
   - *Contradiction for a One-Sided GID:* Generates one 2-dimensional projection.

   Therefore, only a GID with **No Interaction** can be a valid representation.

The projection dimensions act as an unforgeable structural signature. Any valid GID parameterization must adopt a structure that reproduces this signature, thereby forcing its interaction mode to match the intrinsic mode of the subspace. □

**Theorem 5 (Equivalence of SCA and GID for One-Sided Subspaces)** *Let $S$ be a 2D subspace of $\mathbb{R}^{n+c}$ with a **one-sided** or **non-interactive** interaction mode. Its unique orthonormal SCA basis $\{\mathbf{p}^*, \mathbf{q}^*\}$ and the GID basis $\{\mathbf{p}, \mathbf{q}\}$ describe the same subspace, meaning a GID basis $\{\mathbf{p}, \mathbf{q}\}$ can be constructed whose vectors are parallel to the SCA basis vectors $\{\mathbf{p}^*, \mathbf{q}^*\}$, respectively.*

**Proof** The proof is constructive. We will show that for a one-sided SCA basis $\{\mathbf{p}^*, \mathbf{q}^*\}$, we can always find a set of GID primitives $\{\mathbf{u}, \mathbf{v}, \mathbf{z}, \mathbf{r}\}$ with $\|\mathbf{u}\| = 1, \|\mathbf{v}\| = 1$ that generates a GID basis $\{\mathbf{p}, \mathbf{q}\}$ whose vectors are parallel to the SCA basis vectors. Let $S$ be a one-sided subspace. Its orthonormal SCA basis must have one perfectly aligned vector. Without loss of generality, assume this is the c-dominated vector, so $(\mathbf{q}^*)_n = \mathbf{0}$. The SCA basis is:

$$\mathbf{p}^* = \begin{pmatrix} \mathbf{p}_n^* \\ \mathbf{p}_c^* \end{pmatrix}, \quad \text{with } \|\mathbf{p}^*\| = 1$$

$$\mathbf{q}^* = \begin{pmatrix} \mathbf{0} \\ \mathbf{q}_c^* \end{pmatrix}, \quad \text{with } \|\mathbf{q}^*\| = 1 \implies \|\mathbf{q}_c^*\| = 1$$

For the bases to coincide, their vectors must be parallel: $\mathbf{p}^* = c_p \mathbf{p}$ and $\mathbf{q}^* = c_q \mathbf{q}$ for some non-zero scalars $c_p, c_q$. The GID basis vectors are defined as $\mathbf{p} = \begin{pmatrix} \mathbf{u} \\ (\mathbf{u} \cdot \mathbf{z})\mathbf{r} \end{pmatrix}$ and $\mathbf{q} = \begin{pmatrix} (\mathbf{v} \cdot \mathbf{r})\mathbf{z} \\ \mathbf{v} \end{pmatrix}$. **1. Analyzing the ‘q‘ Vector:** The relationship is $\begin{pmatrix} \mathbf{0} \\ \mathbf{q}_c^* \end{pmatrix} = c_q \begin{pmatrix} (\mathbf{v} \cdot \mathbf{r})\mathbf{z} \\ \mathbf{v} \end{pmatrix}$. This yields two conditions:

- $(\mathbf{v} \cdot \mathbf{r})\mathbf{z} = \mathbf{0}$. Since the interaction is one-sided (not non-interactive), $\mathbf{p}_c^* \neq \mathbf{0}$, which will require a non-zero $\mathbf{r}$. To allow for this, we must have the gate $(\mathbf{v} \cdot \mathbf{r}) = 0$. This confirms the GID representation must also be one-sided.
- $\mathbf{q}_c^* = c_q \mathbf{v}$. Taking the norm of both sides and using the GID convention $\|\mathbf{v}\| = 1$ and the SCA property $\|\mathbf{q}_c^*\| = 1$, we get $1 = |c_q| \cdot 1$, so $|c_q| = 1$.

**2. Analyzing the ‘p‘ Vector:** The relationship is $\begin{pmatrix} \mathbf{p}_n^* \\ \mathbf{p}_c^* \end{pmatrix} = c_p \begin{pmatrix} \mathbf{u} \\ (\mathbf{u} \cdot \mathbf{z})\mathbf{r} \end{pmatrix}$. This yields two conditions:

- $\mathbf{p}_n^* = c_p \mathbf{u}$. Taking the norm and using $\|\mathbf{u}\| = 1$, we get $\|\mathbf{p}_n^*\| = |c_p|$.
- $\mathbf{p}_c^* = c_p (\mathbf{u} \cdot \mathbf{z})\mathbf{r}$.

**3. Constructing the GID Primitives:** We can now define a consistent set of primitives. Let’s choose the positive scalars, so $c_q = 1$ and $c_p = \|\mathbf{p}_n^*\|$.

- From $\mathbf{q}_c^* = c_q \mathbf{v}$, we define $\mathbf{v} \triangleq \mathbf{q}_c^*$. This satisfies $\|\mathbf{v}\| = 1$.
- From $\mathbf{p}_n^* = c_p \mathbf{u}$, we define $\mathbf{u} \triangleq \mathbf{p}_n^* / \|\mathbf{p}_n^*\|$. This is well-defined because $\mathbf{p}_n^*$ cannot be the zero vector; otherwise, both $\mathbf{p}^*$ and $\mathbf{q}^*$ would be purely c-space vectors, contradicting the premise that they form a 2D basis with an n-dominated component.
- We are left with one equation, $\mathbf{p}_c^* = \|\mathbf{p}_n^*\| (\mathbf{u} \cdot \mathbf{z})\mathbf{r}$, and two unknowns, $\mathbf{z}$ and $\mathbf{r}$. This reflects a **scaling ambiguity inherent to the interaction channels**, as the coupling term $(\mathbf{u} \cdot \mathbf{z})\mathbf{r}$ is invariant to scaling $\mathbf{z}$ by a non-zero scalar $\alpha$ and $\mathbf{r}$ by $1/\alpha$. We can resolve this ambiguity by making a **canonical** choice. Let us define:

$$\mathbf{r} \triangleq \mathbf{p}_c^*$$

Substituting this in, we need to find a $\mathbf{z}$ such that $\mathbf{p}_c^* = \|\mathbf{p}_n^*\|(\frac{\mathbf{p}_n^*}{\|\mathbf{p}_n^*\|}\cdot\mathbf{z})\mathbf{p}_c^*$. This simplifies to requiring $(\frac{\mathbf{p}_n^*}{\|\mathbf{p}_n^*\|}\cdot\mathbf{z}) = \frac{1}{\|\mathbf{p}_n^*\|}$. A valid choice is:

$$\mathbf{z} \triangleq \frac{\mathbf{p}_n^*}{\|\mathbf{p}_n^*\|^2}$$

We have now constructed a full set of GID primitives with unit-norm $\mathbf{u}$ and $\mathbf{v}$. The GID basis vector $\mathbf{q}$ generated is $\mathbf{q} = [\mathbf{0};\mathbf{v}] = [\mathbf{0};\mathbf{q}_c^*] = \mathbf{q}^*$. The GID basis vector $\mathbf{p}$ generated is $\mathbf{p} = [\mathbf{u};(\mathbf{u}\cdot\mathbf{z})\mathbf{r}] = [\frac{\mathbf{p}_n^*}{\|\mathbf{p}_n^*\|};(\frac{1}{\|\mathbf{p}_n^*\|})\mathbf{p}_c^*] = \frac{1}{\|\mathbf{p}_n^*\|}\mathbf{p}^*$. The generated GID basis $\{\frac{1}{\|\mathbf{p}_n^*\|}\mathbf{p}^*,\mathbf{q}^*\}$ is parallel to the SCA basis $\{\mathbf{p}^*,\mathbf{q}^*\}$ and therefore spans the same subspace. The non-interactive case follows as a simpler version of this logic. Thus, the representations coincide. □

**Theorem 6 (GID Representation from First Principles)** *An orthonormal basis $\{\mathbf{p},\mathbf{q}\}$ for a 2D subspace $S \subset \mathbb{R}^{n+c}$ that satisfies Principle 1, 2, and 3 is uniquely described by a GID. The basis vectors are parallel to vectors of the form:*

$$\mathbf{p}_{gid} = \begin{pmatrix}\mathbf{u}\\ \langle\mathbf{u},\mathbf{z}\rangle\ \mathbf{r}\end{pmatrix} \quad \textit{and} \quad \mathbf{q}_{gid} = \begin{pmatrix}\langle\mathbf{v},\mathbf{r}\rangle\ \mathbf{z}\\ \mathbf{v}\end{pmatrix} \tag{23}$$

*where the primary components $\mathbf{u} \triangleq \mathbf{p}_n/\|\mathbf{p}_n\|$ and $\mathbf{v} \triangleq \mathbf{q}_c/\|\mathbf{q}_c\|$ are unique up to a sign flip. The interaction is governed by a unique rank-1 matrix $\mathbf{Z} \triangleq \mathbf{z}\mathbf{r}^\top$. The individual channel vectors $\{\mathbf{z},\mathbf{r}\}$ are subsequently determined up to a reciprocal scaling ambiguity, where $\{s\mathbf{z}_0,\frac{1}{s}\mathbf{r}_0\}$ for any non-zero scalar $s$ describes the family of valid solutions.*

**Lemma 7 (Optimality Condition from Principle 1)** *An orthonormal basis $\{\mathbf{p},\mathbf{q}\}$ for a 2D subspace satisfies Principle 1 (Minimum Structural Coupling) if and only if its components satisfy the condition:*

$$\langle\mathbf{p}_n,\mathbf{q}_n\rangle = \langle\mathbf{p}_c,\mathbf{q}_c\rangle\ . \tag{24}$$

**Proof** Any orthonormal basis for a 2D subspace can be parameterized by a rotation angle $\theta$, such that $\mathbf{p}(\theta) = \cos(\theta)\mathbf{e}_1 + \sin(\theta)\mathbf{e}_2$ and $\mathbf{q}(\theta) = -\sin(\theta)\mathbf{e}_1 + \cos(\theta)\mathbf{e}_2$ for some fixed basis $\{\mathbf{e}_1,\mathbf{e}_2\}$. The cost function from Principle 1 is $J(\theta) = \|\mathbf{p}_c(\theta)\|^2 + \|\mathbf{q}_n(\theta)\|^2$. We seek the basis that corresponds to the global minimum of this function. First, we find the stationary points of $J$ by setting its derivative of the cost function to zero:

$$\frac{dJ}{d\theta} = \frac{d}{d\theta}\|\mathbf{p}_c\|^2 + \frac{d}{d\theta}\|\mathbf{q}_n\|^2 = 2\left\langle\mathbf{p}_c,\frac{d\mathbf{p}_c}{d\theta}\right\rangle + 2\left\langle\mathbf{q}_n,\frac{d\mathbf{q}_n}{d\theta}\right\rangle$$

Using the facts that $\frac{d\mathbf{p}}{d\theta} = \mathbf{q}$ and $\frac{d\mathbf{q}}{d\theta} = -\mathbf{p}$, we have:

$$\frac{dJ}{d\theta} = 2\langle\mathbf{p}_c,\mathbf{q}_c\rangle - 2\langle\mathbf{q}_n,\mathbf{p}_n\rangle$$

Setting the derivative to zero yields the necessary condition for any extremum: $\langle\mathbf{p}_n,\mathbf{q}_n\rangle = \langle\mathbf{p}_c,\mathbf{q}_c\rangle$. Now, we must confirm that this condition identifies the global minimum required by Principle 1. We can first build an intuition for this. Since the basis vectors $\mathbf{p}(\theta)$ and $\mathbf{q}(\theta)$ are formed by sinusoidal functions of $\theta$, the cost $J(\theta)$ is a sum of squared sinusoidal terms. Using trigonometric identities, this function can be shown to take the general form of a simple, phase-shifted sinusoid, $A + B\cos(2\theta+\phi)$, which is guaranteed to have a unique minimum and maximum over each period. To prove this formally without relying on algebraic expansion, we can use a more general argument. By the **Extreme Value Theorem**, $J(\theta)$ is a continuous function on a compact domain ($[0,2\pi]$) and is therefore guaranteed to attain a global minimum. This global minimum must occur at a stationary point[2] where $\frac{dJ}{d\theta} = 0$. Thus, the global minimum must be among the stationary points of $J$ in the interval of $[0,2\pi]$. Let the basis at an angle $\theta_0$ that satisfies the stationary condition be $\{\mathbf{p},\mathbf{q}\}$. The other extremum occurs at $\theta_0 + \pi/2$, yielding the basis $\{\mathbf{q},-\mathbf{p}\}$. The sum of the cost function at these two points is a constant:

$$J(\theta_0) + J(\theta_0+\pi/2) = (\|\mathbf{p}_c\|^2 + \|\mathbf{q}_n\|^2) + (\|\mathbf{q}_c\|^2 + \|\mathbf{p}_n\|^2) = \|\mathbf{p}\|^2 + \|\mathbf{q}\|^2 = 2$$

Since the sum of the two extremal values is 2, and they cannot be equal (except the trivial case of $J(\theta)$ being a constant function). Therefore, the two extremal values, $J_{min}$ and $J_{max}$, are distinct in any non-trivial case. This guarantees that a unique basis exists which minimizes the cost, and another which maximizes it. Our stationary point condition, $\langle\mathbf{p}_n,\mathbf{q}_n\rangle = \langle\mathbf{p}_c,\mathbf{q}_c\rangle$, finds the locations of both of these bases. **Principle 1**, by definition, is the search for the basis that achieves the global minimum cost. Therefore, we are guaranteed that the optimal basis we seek is among the solutions to the stationary point condition. □

**Lemma 8 (Structure of Rank-1 Maps from Principle 2)** *If a basis $\{\mathbf{p},\mathbf{q}\}$ satisfies Principle 2, its components must have the form*

$$\mathbf{p}_c = \langle\mathbf{p}_n,\mathbf{z}\rangle\,\mathbf{r}$$
$$\mathbf{q}_n = \langle\mathbf{q}_c,\mathbf{r}'\rangle\,\mathbf{z}'$$

*for some vectors $\mathbf{z},\mathbf{z}' \in \mathbb{R}^n$ and $\mathbf{r},\mathbf{r}' \in \mathbb{R}^c$.*

**Proof** Principle 2 states that $\mathbf{p}_c = L_{n\to c}(\mathbf{p}_n)$ where $L_{n\to c}$ is a rank-1 linear map. The image of a rank-1 map is a one-dimensional line, so its output must be proportional to a fixed direction vector $\mathbf{r} \in \mathbb{R}^c$. We can thus write $L_{n\to c}(\mathbf{x}) = f(\mathbf{x})\mathbf{r}$, where $f:\mathbb{R}^n \to \mathbb{R}$ is a scalar function that must be linear. By the Riesz Representation Theorem, any linear functional $f(\mathbf{x})$ can be represented as an inner product $\langle\mathbf{x},\mathbf{z}\rangle$ for some unique vector $\mathbf{z} \in \mathbb{R}^n$. Therefore, $L_{n\to c}(\mathbf{x}) = \langle\mathbf{x},\mathbf{z}\rangle\,\mathbf{r}$.

An identical argument applies to $L_{c\to n}$. The trivial case of a rank-0 map (the zero map) is inherently included in this form by setting $\mathbf{z} = \mathbf{0}$ or $\mathbf{r} = \mathbf{0}$. □

[2]While finding a global optimum on an interval typically requires checking the boundaries, our domain of rotations is topologically a circle. Because our function $J(\theta)$ is smooth and periodic over this domain, any extremum occurring at the parameterization's boundary (e.g., $\theta = 0$) must also have a zero derivative. The search for stationary points is therefore sufficient to find the global minimum.

**Proof of Theorem 6** The proof synthesizes the three principles to derive the GID structure for an orthonormal basis $\{\mathbf{p}, \mathbf{q}\}$.

1. **Implication of Principle 2 (Minimal Rank):** From Lemma 8, the coupling components are given by rank-1 maps, yielding the expressions:
$$\mathbf{p}_c = \langle \mathbf{p}_n, \mathbf{z} \rangle \, \mathbf{r}$$
$$\mathbf{q}_n = \langle \mathbf{q}_c, \mathbf{r}' \rangle \, \mathbf{z}'$$

2. **Implication of Principle 3 (Shared Mechanism):** This principle requires the maps to be transposes, which in outer-product form means $\mathbf{z}'\mathbf{r}'^\top = (\mathbf{r}\mathbf{z}^\top)^\top = \mathbf{z}\mathbf{r}^\top$. For two rank-1 matrices to be equal, their constituent vectors must be collinear. We can therefore unify the representations by defining a single pair of interaction channels $\{\mathbf{z}, \mathbf{r}\}$ without loss of generality:
$$\mathbf{p}_c = \langle \mathbf{p}_n, \mathbf{z} \rangle \, \mathbf{r}$$
$$\mathbf{q}_n = \langle \mathbf{q}_c, \mathbf{r} \rangle \, \mathbf{z}$$
By defining the primary components as $\mathbf{u} \triangleq \mathbf{p}_n / \|\mathbf{p}_n\|$ and $\mathbf{v} \triangleq \mathbf{q}_c / \|\mathbf{q}_c\|$, we arrive at the exact parametric form of the GID.

3. **Implication of Principle 1 (Minimal Coupling):** This principle provides the final constraint. From Lemma 7, the basis that is maximally disentangled must satisfy $\langle \mathbf{p}_n, \mathbf{q}_n \rangle = \langle \mathbf{p}_c, \mathbf{q}_c \rangle$. This ensures that the GID parameterization derived above is not merely a possible structure, but the unique structure of the optimal basis sought by SCA.

4. **Uniqueness:** The optimal basis $\{\mathbf{p}, \mathbf{q}\}$ that minimizes the cost is unique (up to sign flips and degeneracies in the SCA eigenspectrum). This fixes the primary components $\mathbf{u}$ and $\mathbf{v}$. The outer product $\mathbf{Z} = \mathbf{z}\mathbf{r}^\top$ is subsequently fixed by the basis. The remaining degree of freedom is the well-known reciprocal scaling ambiguity in rank-1 matrix factorization, where for any non-zero scalar $s$, the pair $\{s\mathbf{z}, \frac{1}{s}\mathbf{r}\}$ produces the same outer product $\mathbf{Z}$ and thus the same basis vectors.

□

**Proposition 9 (Obliqueness of the GID Basis in Two-Sided Subspaces)** *Let $\{\mathbf{p}, \mathbf{q}\}$ be a basis for a 2D subspace $S \subset \mathbb{R}^{n+c}$ parameterized by GID, where*
$$\mathbf{p} = \begin{pmatrix} \mathbf{u} \\ \langle \mathbf{u}, \mathbf{z} \rangle \, \mathbf{r} \end{pmatrix} \quad \textit{and} \quad \mathbf{q} = \begin{pmatrix} \langle \mathbf{v}, \mathbf{r} \rangle \, \mathbf{z} \\ \mathbf{v} \end{pmatrix}$$
*If the basis $\{\mathbf{p}, \mathbf{q}\}$ represents a two-sided interaction, then its vectors are not orthogonal; that is, $\langle \mathbf{p}, \mathbf{q} \rangle \neq 0$.*

**Proof** The proof proceeds by first deriving a general expression for the dot product of the GID basis vectors and then showing that the necessary conditions for a two-sided interaction require this dot product to be non-zero.

1. **Derivation of the Dot Product.** The dot product of the basis vectors $\mathbf{p}$ and $\mathbf{q}$ is given by the sum of the dot products of their respective components in $\mathbb{R}^n$ and $\mathbb{R}^c$:
$$\langle \mathbf{p}, \mathbf{q} \rangle = \langle \mathbf{p}_n, \mathbf{q}_n \rangle + \langle \mathbf{p}_c, \mathbf{q}_c \rangle$$
Substituting the GID definitions for these components yields:
$$\langle \mathbf{p}, \mathbf{q} \rangle = \langle \mathbf{u}, (\langle \mathbf{v}, \mathbf{r} \rangle)\mathbf{z} \rangle + \langle (\langle \mathbf{u}, \mathbf{z} \rangle)\mathbf{r}, \mathbf{v} \rangle$$
Since $\langle \mathbf{u}, \mathbf{z} \rangle$ and $\langle \mathbf{v}, \mathbf{r} \rangle$ are scalar quantities, they can be factored out of the outer dot products:
$$\begin{aligned} \langle \mathbf{p}, \mathbf{q} \rangle &= (\langle \mathbf{v}, \mathbf{r} \rangle) \langle \mathbf{u}, \mathbf{z} \rangle + (\langle \mathbf{u}, \mathbf{z} \rangle) \langle \mathbf{r}, \mathbf{v} \rangle \\ &= 2(\langle \mathbf{u}, \mathbf{z} \rangle)(\langle \mathbf{v}, \mathbf{r} \rangle) \end{aligned}$$

2. **Implication of the Two-Sided Interaction Condition.** By definition, a GID represents a two-sided interaction if both of its cross-space coupling components are non-zero vectors. A meaningful interaction requires non-zero interaction channels, so we assume $\mathbf{z} \neq \mathbf{0}$ and $\mathbf{r} \neq \mathbf{0}$. The two conditions are:
   (a) $\mathbf{p}_c = (\langle \mathbf{u}, \mathbf{z} \rangle)\mathbf{r} \neq \mathbf{0}$
   (b) $\mathbf{q}_n = (\langle \mathbf{v}, \mathbf{r} \rangle)\mathbf{z} \neq \mathbf{0}$

   For condition (a) to hold, the product of the scalar $\langle \mathbf{u}, \mathbf{z} \rangle$ and the vector $\mathbf{r}$ must be non-zero. Since we assume $\mathbf{r} \neq \mathbf{0}$, this necessarily implies that the gating term must be non-zero: $\langle \mathbf{u}, \mathbf{z} \rangle \neq 0$. Similarly, for condition (b) to hold, since we assume $\mathbf{z} \neq \mathbf{0}$, it necessarily implies that $\langle \mathbf{v}, \mathbf{r} \rangle \neq 0$.

3. **Conclusion.** The dot product of the basis vectors is $\langle \mathbf{p}, \mathbf{q} \rangle = 2(\langle \mathbf{u}, \mathbf{z} \rangle)(\langle \mathbf{v}, \mathbf{r} \rangle)$. As established in step 2, the condition of a two-sided interaction necessitates that both gating terms, $(\langle \mathbf{u}, \mathbf{z} \rangle)$ and $(\langle \mathbf{v}, \mathbf{r} \rangle)$, are non-zero real numbers. The product of two non-zero real numbers is itself non-zero. Therefore, for any GID basis representing a two-sided interaction, it must be that $\langle \mathbf{p}, \mathbf{q} \rangle \neq 0$. The basis vectors are thus necessarily non-orthogonal.

□

**Proposition 10 (Subspace Confinement of Interaction Channels)** *Any minimum-energy GID solution must have its contextual trigger $\mathbf{z}$ in the column space of the input weight matrix, $\mathrm{span}(W_{in})$, and its contextual effect $\mathbf{r}$ in the column space of the output weight matrix, $\mathrm{span}(W_{out})$.*

**Proof** Let the GID reconstruction for the input weights be $W_{in} = \mathbf{u}\boldsymbol{\alpha}^\top + \mathbf{z}(\mathbf{r}^\top\mathbf{v})\boldsymbol{\beta}^\top$. For this equation to hold, $\mathbf{u}$ must lie in $\mathrm{span}(W_{in})$. Now, decompose $\mathbf{z}$ into a component $\mathbf{z}_\parallel \in \mathrm{span}(W_{in})$ and an orthogonal component $\mathbf{z}_\perp$. The reconstruction equation remains unchanged if we replace $\mathbf{z}$ with $\mathbf{z}_\parallel$, as the term $\mathbf{z}_\perp$ is orthogonal to all other terms on the right-hand side and to the left-hand side. However, the objective function becomes $\|\mathbf{z}_\parallel\|^2 + \|\mathbf{z}_\perp\|^2 + \|\mathbf{r}\|^2$. To minimize this objective, $\|\mathbf{z}_\perp\|^2$ must be zero, proving $\mathbf{z}$ must lie entirely in $\mathrm{span}(W_{in})$. A symmetric argument holds for $\mathbf{r}$ and $W_{out}$.

□

**Proposition 11 (Invertibility of the Coefficient Matrix)** *Given the precondition that $W_{in}$ and $W_{out}$ are of full rank (rank 2), the coefficient matrix $C$ must be invertible.*

**Proof** Assume $C$ is singular. Then its columns $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are linearly dependent, meaning $\boldsymbol{\beta} = k\boldsymbol{\alpha}$ for some scalar $k$. Substituting this into the GID equations for $W_{in}$ and $W_{out}$ shows that the column space of both matrices becomes one-dimensional, spanned by a single vector (e.g., $W_{in} = (\mathbf{u} + k(\mathbf{r}^\top\mathbf{v})\mathbf{z})\boldsymbol{\alpha}^\top$). This implies that $\mathrm{rank}(W_{in}) \leq 1$ and $\mathrm{rank}(W_{out}) \leq 1$, which contradicts the initial condition. Therefore, $C$ must be invertible. □

**Proposition 12 (Core GID Constraints)** *For a valid GID factorization to exist, the primitives in the 2D projected space must satisfy the constraints:*

$$\mathbf{z}_{2d}^\top \mathbf{P}\mathbf{r}_{2d} = 1 \quad , \quad \mathbf{z}_{2d}^\top (\mathbf{P}^{-1})^\top \mathbf{r}_{2d} = 1\,, \tag{25}$$

*where,*

$$\mathbf{P} \triangleq \mathbf{R}_n \mathbf{R}_c^{-1}\,. \tag{26}$$

**Proof** Let $s_u = \mathbf{u}_{2d}^\top \mathbf{z}_{2d}$ and $s_v = \mathbf{v}_{2d}^\top \mathbf{r}_{2d}$. The projected GID equations (Eq. 3, 4) can be written compactly as:

$$\mathbf{R}_n = [\mathbf{u}_{2d} \quad s_v\mathbf{z}_{2d}]\, C^\top$$
$$\mathbf{R}_c = [s_u\mathbf{r}_{2d} \quad \mathbf{v}_{2d}]\, C^\top$$

Since $C$ is invertible, we can solve for $C^\top$ and equate the two expressions to yield $\mathbf{P} \triangleq \mathbf{R}_n\mathbf{R}_c^{-1} = [\mathbf{u}_{2d} \quad s_v\mathbf{z}_{2d}]\,[s_u\mathbf{r}_{2d} \quad \mathbf{v}_{2d}]^{-1}$. Therefore, $\mathbf{P}\,[s_u\mathbf{r}_{2d} \quad \mathbf{v}_{2d}] = [\mathbf{u}_{2d} \quad s_v\mathbf{z}_{2d}]$. Equating the columns gives two vector equations:

$$\mathbf{P}(s_u\mathbf{r}_{2d}) = \mathbf{u}_{2d} \tag{27}$$
$$\mathbf{P}\mathbf{v}_{2d} = s_v\mathbf{z}_{2d} \tag{28}$$

Let's first focus on the first equation. By definition, $s_u = \mathbf{u}_{2d}^\top\mathbf{z}_{2d}$. Substitute Eq. 27 into this definition: $s_u = (\mathbf{P}s_u\mathbf{r}_{2d})^\top\mathbf{z}_{2d} = s_u\mathbf{r}_{2d}^\top\mathbf{P}^\top\mathbf{z}_{2d}$. We can assert that $s_u \neq 0$, because if $s_u = 0$, then from Eq. 27, $\mathbf{u}_{2d}$ must be the zero vector. This contradicts the unit-norm constraint on $\mathbf{u}$ ($\|\mathbf{u}\| = 1 \implies \|\mathbf{u}_{2d}\| = 1$). We can therefore safely divide by $s_u$ to obtain the core constraint: $1 = \mathbf{r}_{2d}^\top\mathbf{P}^\top\mathbf{z}_{2d} = \mathbf{z}_{2d}^\top\mathbf{P}\mathbf{r}_{2d}$. We now switch to the second equation. Substitute Eq. 28 into the definition of $s_v$: $s_v = \mathbf{v}_{2d}^\top\mathbf{r}_{2d} = (\mathbf{P}^{-1}s_v\mathbf{z}_{2d})^\top\mathbf{r}_{2d} = s_v(\mathbf{z}_{2d}^\top(\mathbf{P}^{-1})^\top\mathbf{r}_{2d})$. Since $s_v \neq 0$, we get the second, essential constraint: $\mathbf{z}_{2d}^\top(\mathbf{P}^{-1})^\top\mathbf{r}_{2d} = 1$. □

**Proposition 13** *By solving the two scalar constraints $\mathbf{z}_{2d}^\top\mathbf{P}\mathbf{r}_{2d} = 1$ and $\mathbf{z}_{2d}^\top(\mathbf{P}^{-1})^\top\mathbf{r}_{2d} = 1$ in $\mathbf{z}_{2d}$ and then substituting this solution back into the energy function, we obtain the following unconstrained optimization task:*

$$\min_{\mathbf{r}_{2d}\in\mathbb{R}^2} E(\mathbf{r}_{2d}) = \frac{\mathbf{r}_{2d}^\top(\mathbf{A}^\top\mathbf{A})\mathbf{r}_{2d}}{(\mathbf{r}_{2d}^\top\mathbf{M}\mathbf{r}_{2d})^2} + \mathbf{r}_{2d}^\top\mathbf{r}_{2d} \tag{29}$$

*where $\mathbf{A} = \mathbf{P} - (\mathbf{P}^{-1})^\top$ and $\mathbf{M} = \mathbf{A}^\top\mathbf{J}^\top\mathbf{P}$.*

**Proposition 14 (The Collapsed 1D Objective Function)** *The minimum energy, as a function of the direction vector $\hat{\mathbf{u}} \in \mathbb{S}^1$, is given by:*

$$E(\hat{\mathbf{u}}) = 2 \cdot \frac{\sqrt{\hat{\mathbf{u}}^\top(\mathbf{A}^\top\mathbf{A})\hat{\mathbf{u}}}}{|\hat{\mathbf{u}}^\top\mathbf{M}\hat{\mathbf{u}}|} \tag{30}$$

**Proof** Let $N(\hat{\mathbf{u}}) = \hat{\mathbf{u}}^\top(\mathbf{A}^\top\mathbf{A})\hat{\mathbf{u}}$ and $D(\hat{\mathbf{u}}) = \hat{\mathbf{u}}^\top\mathbf{M}\hat{\mathbf{u}}$. From the previous proposition, the optimal squared magnitude is $\rho^2 = \frac{\sqrt{N(\hat{\mathbf{u}})}}{|D(\hat{\mathbf{u}})|}$. Substitute this into the energy function $E = \rho^{-2}\frac{N(\hat{\mathbf{u}})}{D(\hat{\mathbf{u}})^2} + \rho^2$. The first term becomes $\frac{|D(\hat{\mathbf{u}})|}{\sqrt{N(\hat{\mathbf{u}})}}\frac{N(\hat{\mathbf{u}})}{D(\hat{\mathbf{u}})^2} = \frac{\sqrt{N(\hat{\mathbf{u}})}}{|D(\hat{\mathbf{u}})|}$. The second term is $\rho^2 = \frac{\sqrt{N(\hat{\mathbf{u}})}}{|D(\hat{\mathbf{u}})|}$. Summing these two identical terms gives the final expression. □

**Proposition 15** *A 2D subspace $S$ exhibiting a truly one-sided interaction does not produce an admissible neuron pair for any basis $\{\mathbf{w}_i, \mathbf{w}_j\}$ spanning $S$.*

**Proof** A 2D subspace $S$ is defined as one-sided if the dimension of one of its projections is 1 while the other is 2. Let us consider the case where $\dim(S_n) = 1$ and $\dim(S_c) = 2$. The input weight matrix is formed by the n-space components of the basis vectors, $W_{in} = [(\mathbf{w}_i)_n|(\mathbf{w}_j)_n]$. The columns of $W_{in}$ must both lie within the one-dimensional subspace $S_n$. As a result, the two columns are necessarily collinear, meaning the column space of $W_{in}$ is one-dimensional. Therefore, $\mathrm{rank}(W_{in}) = 1$, which violates the admissibility condition that $\mathrm{rank}(W_{in}) = 2$. □

**Theorem 16** *The exact analytical GID factorization is mathematically undefined for one-sided interaction spaces.*

**Proof** The derivation of the analytical solution breaks down at three sequential stages:

1. **Singularity of the R Matrix.** The derivation begins with the (thin) QR decomposition, such as $W_{in} = \mathbf{Q}_n\mathbf{R}_n$. For a matrix $W_{in} \in \mathbb{R}^{n\times 2}$ with $n \geq 2$, it is a standard result of linear algebra that $\mathrm{rank}(W_{in}) = \mathrm{rank}(\mathbf{R}_n)$. For a one-sided space with $\mathrm{rank}(W_{in}) = 1$, the resulting $2 \times 2$ upper-triangular matrix $\mathbf{R}_n$ must also have a rank of 1. A square matrix with deficient rank is, by definition, singular, meaning $\det(\mathbf{R}_n) = 0$ and its inverse $\mathbf{R}_n^{-1}$ does not exist.

2. **Ill-definition of the Transformation Matrix $\mathbf{P}$.** The central object in the derivation is the $2 \times 2$ matrix $\mathbf{P} \triangleq \mathbf{R}_n \mathbf{R}_c^{-1}$. In the case of a one-sided space where $\text{rank}(W_{in}) = 1$ and $\text{rank}(W_{out}) = 2$, the matrix $\mathbf{R}_c$ is invertible, but $\mathbf{R}_n$ is singular. While $\mathbf{P}$ can be formed, it will itself be singular. Crucially, its inverse, $\mathbf{P}^{-1}$, which is required for the second constraint, is undefined.

3. **Collapse of the Constraint System.** The entire analytical solution hinges on the existence of two independent constraints derived from the GID structure:
$$\mathbf{z}_{2d}^\top \mathbf{P} \mathbf{r}_{2d} = 1$$
$$\mathbf{z}_{2d}^\top (\mathbf{P}^{-1})^\top \mathbf{r}_{2d} = 1$$
Since $\mathbf{P}^{-1}$ does not exist, the second constraint is mathematically meaningless. The system of equations that defines the relationship between the interaction channels is therefore incomplete and underdetermined, and the analytical path to the quartic polynomial is broken at its foundation.

Thus, the formulation is rigorously undefined for any one-sided interaction space. □

**Proposition 17 (Structural Cost as an Upper Bound on Functional Impact)** *Let $L_{D_{val}}(\mathbf{w}_i, \mathbf{w}_j, \mathbf{w}_{rest})$, the network's loss function evaluated over a validation set, be $\mathscr{L}_{D_{val}}$-Lipschitz continuous with respect to the weights $\mathbf{w}_{ij} \triangleq \begin{bmatrix} \mathbf{w}_i \\ \mathbf{w}_j \end{bmatrix}$. The change in loss, $|\Delta L_{D_{val}}|$, i.e. $C_{ideal}(i,j)$, resulting from merging a neuron pair $(i,j)$ is bounded by the true structural cost:*
$$|\Delta L_{D_{val}}| \leq \mathscr{L}_{D_{val}} \sqrt{C_{struct}(i,j)}$$

**Proof** For brevity, we drop the subscript $D_{val}$ in the loss function's notation. The merge operation changes the network parameters from $\mathbf{w}_{\text{all}}$ to $\mathbf{w}'_{\text{all}} = \mathbf{w}_{\text{all}} + \Delta\mathbf{w}_{\text{all}}$. The perturbation $\Delta\mathbf{w}_{\text{all}}$ is non-zero only for the weights of neurons $i$ and $j$, where $\Delta\mathbf{w}_i = \mathbf{w}_{\text{new}} - \mathbf{w}_i$ and $\Delta\mathbf{w}_j = -\mathbf{w}_j$. By definition of Lipschitz continuity, $|\Delta L| = |L(\mathbf{w}'_{\text{all}}) - L(\mathbf{w}_{\text{all}})| \leq \mathscr{L} \|\Delta\mathbf{w}_{\text{all}}\|$. The squared norm is $\|\Delta\mathbf{w}_{\text{all}}\|^2 = \|\Delta\mathbf{w}_i\|^2 + \|\Delta\mathbf{w}_j\|^2 = \|\mathbf{w}_{\text{new}} - \mathbf{w}_i\|^2 + \|-\mathbf{w}_j\|^2 = C_{\text{struct}}(i,j)$. Thus, $|\Delta L| \leq \mathscr{L}\sqrt{C_{\text{struct}}(i,j)}$. □

**Proposition 18 (Perfect Reversibility of the Un-Merge Operation)** *The un-merge operation is the exact algebraic inverse of the GID factorization performed during encoding. Given an instruction set $I_{ij}$, the decoder can perfectly reconstruct the original neuron pair $(\mathbf{w}_i, \mathbf{w}_j)$.*

**Proof** Let the encoder map $(\mathbf{w}_i, \mathbf{w}_j) \to (\mathbf{w}_{\text{new}}, I_{ij})$. The decoder receives the instruction packet $I_{ij}$ and the current state of the simplified network. The reconstruction for each child neuron $h \in \{i, j\}$ proceeds in four deterministic steps, using only the information in $I_{ij}$:

1. **Direct Recovery of Primitives:** The decoder retrieves the *complete* set of original GID primitives $\{\mathbf{u}, \mathbf{v}, \mathbf{z}, \mathbf{r}\}$ and reconstruction coefficients $\{\alpha_h, \beta_h\}$ directly from the instruction packet $I_{ij}$. This direct retrieval protocol is robust and avoids the numerical instability of inferring the primary vectors from the parent neuron's weights.

2. **Algebraic Reconstruction of Effective Weights:** Using the recovered primitives, the decoder algebraically reconstructs the *effective* weights of the child neuron, $(\mathbf{w}_h)_{\text{eff}}$, which represent the weights after Group Normalization scaling has been applied.
$$(\mathbf{w}_h)_{\text{eff}}^{\text{in, rec}} = \alpha_h \mathbf{u} + \beta_h \langle \mathbf{v}, \mathbf{r} \rangle \mathbf{z} = (\mathbf{w}_h)_{\text{eff}}^{\text{in}}$$
$$(\mathbf{w}_h)_{\text{eff}}^{\text{out, rec}} = \beta_h \mathbf{v} + \alpha_h \langle \mathbf{u}, \mathbf{z} \rangle \mathbf{r} = (\mathbf{w}_h)_{\text{eff}}^{\text{out}}$$

3. **Reversal of Group Normalization:** To obtain the final *bare* kernel weights that are stored in the model, the decoder must reverse the effect of the Group Normalization layer.[3] It retrieves the original scalar GN parameter $\gamma_h$ from the instruction packet and performs the inverse scaling operation:
$$(\mathbf{w}_h)_n^{\text{bare}} = \frac{(\mathbf{w}_h)_{\text{eff}}^{\text{in, rec}}}{\gamma_h}$$
The c-space weights, $(\mathbf{w}_h)_c$, are already bare and require no modification.

4. **Residual Correction for Numerical Precision:** Finally, the decoder adds the stored numerical residual vector $\Delta\mathbf{w}_h$ from the instruction packet back to the reconstructed weights.
$$\mathbf{w}_h^{\text{final}} = \mathbf{w}_h^{\text{rec}} + \Delta\mathbf{w}_h$$
This step corrects for any minuscule floating-point errors from the encoder's GID factorization, ensuring the final result is bit-for-bit identical to the original weight vector $\mathbf{w}_h$.

After computing the final child neuron weights, the decoder updates the simplified network by replacing the parent neuron with the two reconstructed child neurons. Since all primitives are retrieved directly and all architectural transformations are explicitly reversed, the un-merge operation is a perfect reversal. □

**Proposition 19 (Propagation of Elasticity to All Trainable Parameters)** *A neuron's function is determined by the totality of its trainable parameters, not just its kernel weights. To maintain functional coherence during fine-tuning, all parameters associated with a given neuron must share the same degree of plasticity.*

**Proof by Contradiction** Assume the kernel weights of a neuron are assigned a high elasticity, but its associated bias and Group Normalization ('beta' and 'gamma') parameters are frozen (zero elasticity). During fine-tuning, the optimizer can change the kernel, altering the neuron's feature selectivity. However, it cannot adjust the output statistics (mean and variance) of the neuron's activations, as the bias, shift ('beta'), and scale ('gamma') are fixed. The new feature distribution produced by the updated kernel may be poorly conditioned for subsequent layers, but the network lacks the local degrees of freedom to correct it. This functional inconsistency can impair learning. Therefore, for effective adaptation, all trainable parameters of a neuron must be allowed to co-adapt with the same degree of freedom. □

[3] The other learnable parameter of Group Normalization, the additive shift $\delta_h$, is not part of this multiplicative reversal. It is treated as a simple bias term that is not included in the GID factorization. Instead, its original value is archived in the instruction packet by the encoder and restored directly by the decoder, separate from this weight reconstruction process.