LEARNING ABSTRACT DISENTANGLED FACTORS FROM ENTANGLED CONJUNCTIVE CODES THROUGH COMPOSITIONAL TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Abstract, or disentangled, representations in neural networks encode latent factors in separate subspaces and can support compositional generalization. Yet latent factors are often observed only through entangled combinations, posing a challenge for networks to disentangle them. Here we study compositional tasks whose inputs and outputs consist of conjunctions of discrete latent factors (e.g., shape $\times$ color). Each combination of factor values in a conjunction (e.g., *square* $\times$ *blue*) receives a distinct one-hot code. We find that even when a latent factor X is always entangled with other factors in the input and the output, it can become abstract in task-trained networks through **rebinding**. Rebinding occurs when X is *bound* to (i.e. encoded in a conjunction with) one factor in the input and a different factor in the output. We illustrate this mechanism in a rule transfer task inspired by Raven's Progressive Matrices, for which we analytically derive the optimal hidden representation of single-hidden-layer ReLU networks. The solution uses subpopulations of mixed-selective neurons, each supporting abstraction for a subset of factor values. We then derive the optimal linear network representation for arbitrary tasks in the family. The theory formalizes the rebinding principle and predicts that abstraction increases when a factor is reused across many distinct conjunctions. The predicted degree of abstraction qualitatively matches that observed in nonlinear networks and closely tracks their generalization performance to held-out contexts in a transfer task. Together, these results show how compositional task structure can drive networks to learn abstract representations from entangled observations.

1 INTRODUCTION

A hallmark of intelligence is the ability to apply learned knowledge to new situations. This flexibility can be supported by *abstract* or *disentangled* representations, where the population-level encoding of one variable remains invariant as other variables change, allowing the same information to be reused across contexts (i.e. sets of values of the other variables). Such geometries can support few-shot and compositional generalization in neural networks (Ito et al., 2022; Johnston & Fusi, 2023; Vafidis et al., 2025), and have been observed in the brain during flexible behavior (Bernardi et al., 2020; Courellis et al., 2024). We use *abstract* and *disentangled* interchangeably to refer specifically to this population-level invariance: individual neurons need not correspond to individual factors.

A central challenge for learning abstract representations is that latent factors are often observed only through combinations with other factors. For example, image pixels jointly depend on object identity, pose, viewpoint, and illumination. A learner observes only their combined effect, and observations that share a factor value need not be similar in any obvious way. Recovering factors therefore requires additional assumptions or supervision signals (Locatello et al., 2019). Here we ask whether a compositional input–output mapping can provide such a signal, allowing abstract representations of individual factors to emerge from entangled inputs without directly labeling those factors or identifying which observations share them.

We start with a rule transfer task inspired by Raven's Progressive Matrices (RPMs), a classic test of flexible reasoning (Raven, 1938). Rather than modeling the full complexity of RPMs, we isolate a

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

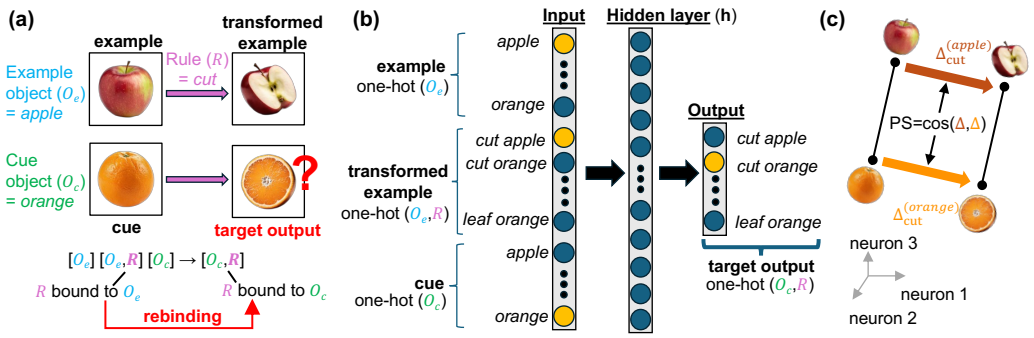


Figure 1: **Rule transfer task. (a)** An RPM-inspired task requires extracting a rule from an example object and applying it to a cue object. The rule changes its binding partner between input and output. **(b)** Input/output encoding and network architecture. Each item is represented by a one-hot code; conjunctions sharing a rule have no built-in similarity. **(c)** We quantify abstraction by the parallelism score (PS), the cosine similarity between coding directions for the same variable across contexts (i.e. values of other variables).

basic computational motif: extracting a reusable rule from one context and applying it in another. For example, in Fig. 1a, the first row *apple* and *cut apple* specify the rule *cut*, which must then be combined with a new cue object, *orange*, to produce the correct output *cut orange*. Sharing the same rule need not imply any apparent similarity at the pixel level (e.g. the interiors of a cut apple and a cut orange can differ substantially in color and texture). To make this maximally challenging, we assign a distinct one-hot code to every conjunction value, so e.g., *cut apple* and *cut orange* are orthogonal.

Leveraging recently developed tools for analyzing feature learning in regularized two-layer nonlinear networks (Sahiner et al., 2020; Jacot et al., 2022; Wang et al., 2026), we derive the optimal hidden representation of ReLU networks for this task. The optimal solution develops abstract rule representations. The key task structure that promotes abstraction is **rebinding**: the rule is *bound* to (i.e., encoded in conjunction with) the example object in the input, but to a different factor (cue object) in the output. In this task, the network discards the example object identity as it is irrelevant to the output, making rule coding invariant across example objects. Rule and cue object remain partially coupled because their conjunction is required at the output, and the theory predicts their exact degree of abstraction (quantified using the parallelism score, introduced in Bernardi et al. (2020), illustrated in Fig. 1c, and discussed in detail below). The theory also predicts the tunings of individual neurons, which organize into distinct subpopulations with *rectangular* tuning: each neuron responds uniformly to all combinations of a particular subset of rules and a particular subset of cue objects. Different subpopulations are therefore perfectly abstract for specific variations of factor values but not others. This structured mixed selectivity allows the population to retain the rule–cue interaction needed for the conjunctive output while preserving partial abstraction.

The rule transfer task still presents example and cue objects individually. We therefore consider a more challenging recombination task in which *no* latent factor is presented alone: every input and output item is a conjunction of multiple factors. Nevertheless, trained networks develop similar abstract representations and neuron tunings. To understand why such representations emerge more generally, we derive the optimal hidden representation of regularized linear networks for this family of conjunctively encoded tasks. The theory formalizes rebinding and reveals a graded extension: abstraction increases as a factor is reused across more conjunctions with distinct partners, in either the input or the output. Nonlinear ReLU networks qualitatively follow these predictions across a large task bank. Moreover, the degree of abstraction predicted from task structure closely tracks generalization to held-out contexts in a downstream transfer task. These results provide a predictive account of how the combinatorial structure of a task can transform entangled observations into abstract, reusable representations.

2 Related work

Learning abstraction from structured supervision One approach to learning abstract representations is to train networks on multiple tasks involving the same latent variables, such as classification tasks with different linear boundaries in latent space (Johnston & Fusi, 2023; Vafidis et al., 2025; Wakhloo et al., 2026). Related theory shows that task diversity can enable recovery of shared latent factors up to linear transformation (Chen et al., 2024) and reduce the data needed to learn new tasks (Tripuraneni et al., 2020). Another approach exploits structured observations, for example by grouping observations that share a factor (Bouchacourt et al., 2018), or pair observations in which some factors change while others remain fixed (Locatello et al., 2020). Our setting differs in that individual latent factors are never directly identified by the observations. Their occurrence is always embedded in one-hot conjunctions, and conjunctions sharing a factor can be orthogonal. The latent structure must therefore be inferred from how these conjunctions participate in the input–output mapping.

Network theories of compositional representation and generalization Several works characterize how latent task structure shapes learned representation geometry. Alleman et al. (2024) shows that input geometry, target geometry, and activation nonlinearity jointly shape hidden representations. Analyses of linear networks (Yang et al., 2026; Jarvis et al., 2026; Riveland et al., 2026) on compositionally structured tasks reveal the effects of depth and initialization. Other works (Johnston & Fusi, 2024; Dorrell et al., 2025) focus on conditions when the optimal representations become modular. Lippl & Stachenfeld (2025) studied how fixed compositional representations constrain generalization to unseen combinations. Shang et al. (2026) predicts generalization performance to held-out rules from properties of the learned representation in a simplified RPM task. Most closely, Wang et al. (2026) developed a measure-space optimization framework to determine representations learned in nonlinear networks when outputs depend directly on individual latent variables. We extend this line of work to settings where latent factors are entangled in both input and output and characterize the resulting optimal representations.

3 Rule rebinding induces abstraction in an RPM-inspired rule transfer task

3.1 Setup

Rule transfer task We study a minimal rule transfer task inspired by RPMs (Fig. 1a-b). Let $\mathcal{O} = \{O_1, \dots, O_n\}$ and $\mathcal{R} = \{R_1, \dots, R_m\}$ denote possible object and rule values, with $m, n \geq 2$. Each trial is specified by an example object $O_e \in \mathcal{O}$, a cue object $O_c \in \mathcal{O}$, and a rule $R \in \mathcal{R}$. O_e and O_c need not differ. The task maps an example object, its rule-transformed version, and a cue object to the transformed cue:

$$[O_e],\ [O_eR],\ [O_c] \quad \longrightarrow \quad [O_cR]. \tag{1}$$

Each item is encoded by a separate one-hot code. Writing $\mathbf{e}_i^{(d)}$ for the i-th standard basis vector in $\mathbb{R}^d$, the task input and output vectors are

$$\mathbf{x}(O_i, R_j, O_k) = \mathbf{e}_i^{(n)} \oplus \mathbf{e}_{(i,j)}^{(mn)} \oplus \mathbf{e}_k^{(n)}, \qquad \mathbf{y}(O_i, R_j, O_k) = \mathbf{e}_{(k,j)}^{(mn)}. \tag{2}$$

where $\oplus$ denotes concatenation and (i, j) indexes the mn object–rule conjunction values (Fig. 1b).

Network We consider one-hidden-layer ReLU networks trained on all $P = mn^2$ trials. The hidden activity is $\mathbf{h}(\mathbf{x}) = \phi(W_1\mathbf{x} + \mathbf{b}_1)$ and the output is $\hat{\mathbf{y}}(\mathbf{x}) = W_2\mathbf{h}(\mathbf{x}) + \mathbf{b}_2$, where ϕ is element-wise ReLU. Let $X_{\text{data}} = [\mathbf{x}_1, \dots, \mathbf{x}_P]$, $Y = [\mathbf{y}_1, \dots, \mathbf{y}_P]$ and $H = [\mathbf{h}(\mathbf{x}_1), \dots, \mathbf{h}(\mathbf{x}_P)]$, with trials as columns. We minimize squared error with L_2 regularization:

$$\mathcal{L} = \|Y - W_2H - \mathbf{b}_2\mathbf{1}_P^\top\|_F^2 + \lambda\big(\|W_1\|_F^2 + \|\mathbf{b}_1\|_2^2 + \|W_2\|_F^2 + \|\mathbf{b}_2\|_2^2\big). \tag{3}$$

Our theory characterizes global optima for sufficiently wide networks in the limit $\lambda \downarrow 0$ (Appendix C.1). Simulation details are in Appendix B.

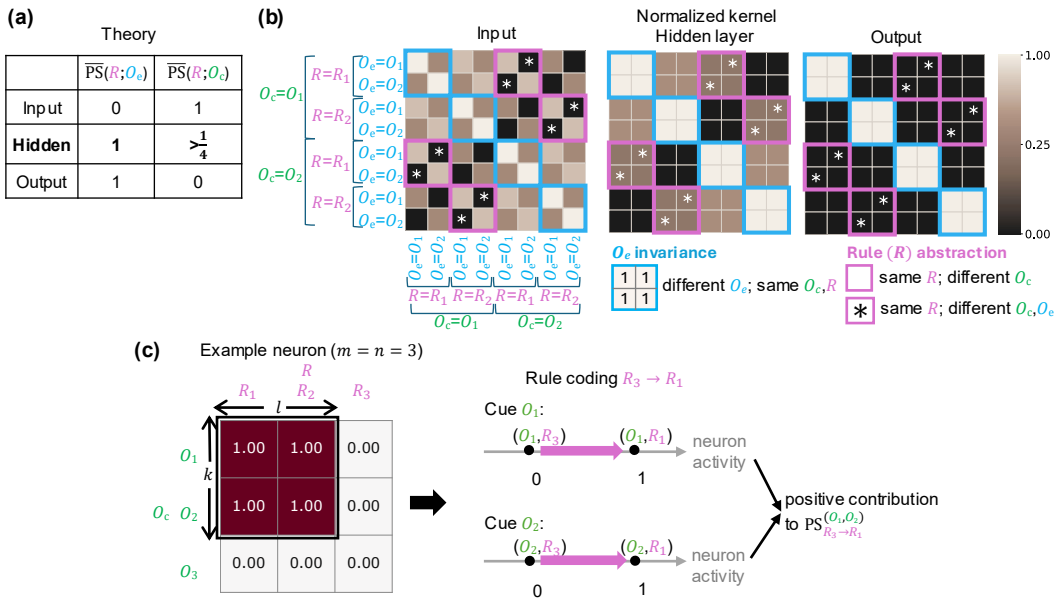


Figure 2: **Abstract rule geometry and rectangular neuronal tuning in rule transfer task. (a)** Theoretical marginal rule PS. **(b)** Input, hidden, and output kernels for $m = n = 2$, scaled to unit diagonal. Trials (rows and columns) are ordered by O_c, then R, then O_e. Starred entries highlight similarities that emerge in the hidden layer despite being absent from input and output. **(c)** Example rectangular neuron for $m = n = 3$. Activity is averaged over O_e and scaled to unit maximum.

Parallelism Score (PS) We quantify abstraction using the PS, which measures the alignment of a target variable's coding directions across contexts. For example, in Fig. 1c, the geometry has high PS as the coding direction of the rule *cut* for the object *apple* is parallel to the coding direction of *cut* for *orange* Bernardi et al. (2020)). This geometry reflects a process of abstraction because in the subspace aligned to these coding directions, the rule *cut* does not depend on the object (i.e. it is disassociated from the specific object instances). We adapt PS to our tasks using a *marginal* PS (Fig. S1), defined for a target factor and a context factor. For example, to compute the marginal PS of the rule R with respect to the cue object O_c, we first marginalize the representation over the remaining factor O_e: $\bar{\mathbf{h}}(R, O_c) = \frac{1}{n}\sum_{O_e \in \mathcal{O}} \mathbf{h}(O_e, R, O_c)$. For two rule values R_i, R_j and two cue-object contexts O_k, O_ℓ, we define the pair-specific PS as

$$\mathrm{PS}^{(k,\ell)}_{i,j} = \cos\left(\Delta^{(k)}_{i\to j}, \Delta^{(\ell)}_{i\to j}\right), \qquad \Delta^{(k)}_{i\to j} = \bar{\mathbf{h}}(R_j, O_k) - \bar{\mathbf{h}}(R_i, O_k). \tag{4}$$

where $\Delta^{(k)}_{i\to j}$ is the rule-coding direction in cue-object context O_k. We average over all pairs of rule and cue-object values to obtain the marginal PS: $\overline{\mathrm{PS}}(R; O_c) = \left\langle \mathrm{PS}^{(k,\ell)}_{i,j} \right\rangle_{i<j,\ k<\ell}$. A marginal PS close to 1 indicates the coding direction of the target variable is consistent across different values of the context variable, corresponding to an abstract representation. In contrast, a marginal PS close to 0 indicates strong dependence of the target variable coding on the context, corresponding to an entangled, non-abstract representation.

3.2 Rebinding induces abstract rule representations

Although the rule is never presented in isolation and rule–object conjunctions sharing the same rule have no built-in similarity, the optimal hidden representation develops an abstract rule code. We identify two notable properties shared by all global optima of the network trained on the above rule transfer task (Eqs. 1-3, Appendix C.1).

First, every global optimum discards example-object identity:

$$\mathbf{h}(O_e, R, O_c) = \mathbf{h}(O'_e, R, O_c), \qquad \text{for all } O_e, O'_e, R, O_c. \tag{5}$$

Discarding the output-irrelevant example object reduces the weight cost. Rule coding is therefore fully invariant to the example object, giving $\overline{\mathrm{PS}}(R; O_e) = 1$ (Fig. 2a).

Second, rule coding cannot become fully invariant across cue objects, because the linear readout must still recover the correct rule–cue conjunction at the output. Nevertheless, the optimal hidden representation remains partially abstract:

$$\overline{\mathrm{PS}}(R; O_c) \geq 2 - \sqrt{3} > \frac{1}{4}, \qquad \text{for all } m, n \geq 2. \tag{6}$$

The theory further gives exact PS predictions (Appendix C.1), which closely match numerical results across task sizes (Fig. S2).

This selective abstraction follows the task's change of binding. At the input, the rule is bound to the example object but independent of the cue object, giving $\overline{\mathrm{PS}}(R; O_e) = 0$ and $\overline{\mathrm{PS}}(R; O_c) = 1$. At the output, the rule is instead bound to the cue object and independent of the example object, giving $\overline{\mathrm{PS}}(R; O_e) = 1$ and $\overline{\mathrm{PS}}(R; O_c) = 0$. The hidden layer therefore removes the old rule–example binding while partially retaining the new rule–cue binding required for the output.

This geometry is directly visible in the hidden kernel $K_H = H^\top H$: trials differing only in the example object have identical hidden representations (uniform blue blocks in Fig. 2b middle). More strikingly, trials that share the same rule but differ in both the example and cue objects acquire positive similarity in the hidden kernel, despite zero similarity in the input and output (starred cells).

3.3 Rectangular mixed-selective neuron subpopulations

Although the population representation is abstract, individual neurons remain mixed-selective rather than encoding single factors (for $m, n \geq 3$, Appendix C.1). Specifically, each neuron has *rectangular tuning*: neuron a responds uniformly to a subset $G_a \subseteq \mathcal{O}$ of cue objects and a subset $S_a \subseteq \mathcal{R}$ of rules, and is silent elsewhere:

$$h_a(O_e, R, O_c) = d_a \, \mathbf{1}\{O_c \in G_a\}\mathbf{1}\{R \in S_a\}, \qquad d_a > 0. \tag{7}$$

This tuning enables conjunctive readouts while generating partial abstraction. For example, the neuron in Fig. 2c has $G_a = \{O_1, O_2\}$ and $S_a = \{R_1, R_2\}$. Consider the rule contrast $R_3 \to R_1$ in cue-object contexts O_1 and O_2. In both contexts, this neuron's activity changes from 0 to $d_a = 1$, thus providing the same coordinate to the rule-coding vectors $\Delta^{(O_1)}_{R_3 \to R_1}$ and $\Delta^{(O_2)}_{R_3 \to R_1}$ and contributing positively to $\mathrm{PS}^{(O_1, O_2)}_{R_3 \to R_1}$ (Eq. 4). More generally, computing PS within a specific subpopulation with tuning support $G_a \times S_a$ yields $\mathrm{PS}^{(O_u, O_v)}_{R_q \to R_p} = 1$ when $O_u, O_v \in G_a$ and exactly one of R_p, R_q belongs to S_a. Thus different neural subpopulations exhibit abstraction for different pairs of rules and contexts.

For any rectangle size $k = |G_a|$ and $\ell = |S_a|$ active at the optimum, there is an optimal solution containing all choices of G_a and S_a of those sizes (Fig. S3). The corresponding subpopulation has marginal rule PS $\overline{\mathrm{PS}}(R; O_c) = \frac{k-1}{n-1}$. Active rectangle sizes typically lie near $k \approx n/2$, giving PS near $1/2$.

4 Abstraction extends to a compositional recombination task

In the rule transfer task, the example object and the cue object are already presented as single-factor components in the input, and discarding information about the output-irrelevant example object makes the transformed example depend only on the rule. We therefore ask whether abstraction can emerge when no factor is presented alone, and removing irrelevant factors is not sufficient to expose the individual factors.

One setting with this structure is recombination of familiar components (Fig. 3a). For example, we may be familiar with the taste of *orange juice* and *dried apple chips*, and can imagine swapping their preparation methods (*dried orange chips* and *apple juice*). The serving plate could be an irrelevant factor for the taste. Denoting the two ingredient identities by A and D, their preparation methods by B and C, and the irrelevant plate by E, the task is

$$[AB], [CDE] \to [AC], [BD]. \tag{8}$$

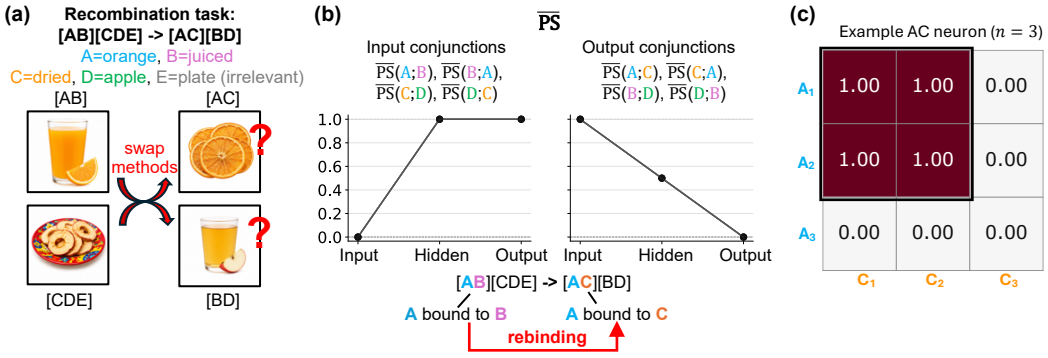


Figure 3: **Abstraction and rectangular tuning extend to a recombination task in simulations.** **(a)** Example recombination task. **(b)** Marginal PS at the input, hidden layer, and output for pairs of output-relevant factors whose conjunctions are in the input or output (no conjunction is in both; conjunctions not in either correspond to factor pairs already abstract in the input and the output). Within each group, the four PS values are very similar and indistinguishable in the plot. **(c)** Example AC-selective neuron showing rectangular tuning. **(b)–(c)** use $n = 3$.

Here $A, \ldots, E$ are categorical factors, each with n possible values. Each bracket denotes one component with a one-hot code for the included factors: for example, $[AB] \in \mathbb{R}^{n^2}$ is a one-hot code for the conjunction of the values of A and B. Multiple components are concatenated in the input and output. In this notation, the rule transfer task is $[A], [AB], [C] \rightarrow [BC]$, with $(A, B, C) = (O_e, R, O_c)$.

In the recombination task (Eq. 8), even after discarding the irrelevant E, the four relevant factors A, B, C, D appear only in conjunctions. However, each is rebound: e.g., A is bound to B in the input but to C in the output, so the same rebinding mechanism may still promote abstraction. Indeed, simulations of ReLU networks produce abstraction patterns similar to the rule transfer task (Fig. 3b). Factor pairs bound only in the input (A–B and C–D), which have PS 0 in the input and 1 in the output, become almost perfectly abstract in the hidden layer. Pairs bound together at the output (A–C and B–D), which have PS 1 in the input and 0 in the output, are partially abstract in the hidden layer, retaining some conjunctive structure for readout. Pairs bound at neither (A–D and B–C) are already abstract in the input and output, and remain nearly fully abstract in the hidden layer (Fig. S4b).

The learned kernel also reveals the emergent geometry of the task-relevant factors (Fig. S4a). Trials differing only in E become nearly indistinguishable, showing that the irrelevant factor is discarded (blue boxes). Meanwhile, for each relevant factor, trials sharing only that factor (e.g. A) but differing in the other three (e.g. B, C, D) acquire positive similarity in the hidden kernel, despite zero similarity in the raw input and output (magenta boxes show example for A in Fig. S4a).

At the neuron level, active neurons in the $n = 3$ simulation form two subpopulations selective for the AC and BD conjunctions respectively, matching the two output conjunctions (Fig. S4c-d). Within each subpopulation, tuning is approximately invariant to the other factors: e.g. activity of an AC neuron is nearly constant across B, D variations (Fig. S4c). For the two factors a neuron is tuned to, it shows the same rectangular tuning as seen in the rule transfer task (Fig. 3c and S4c). Each subpopulation contains all nine possible 2×2 rectangle subtypes (Fig. S4d). Together, these simulations show that the rebinding-driven abstraction and structured neuronal tuning extend beyond the rule transfer task, even when every factor is observed only through conjunctions.

5 A LINEAR THEORY LINKS TASK STRUCTURE TO ABSTRACTION

Why do similar abstraction patterns emerge in the rule transfer and recombination tasks, and can we predict them for other compositional tasks? To make the theory more tractable for arbitrary tasks in this family, we turn to regularized networks with one *linear* hidden layer. Although linear networks cannot always produce the correct conjunctive output from the input, their optimal representations

provide explicit predictions about which factors become abstract, giving us valuable intuitions that we test in nonlinear networks in Section 6.

5.1 Optimal linear representation

We consider tasks with d categorical factors $\{A, B, C, ...\}$, each taking n values (Appendix C.2). The input and output are specified by collections of factor subsets, $\mathcal{I}$ and $\mathcal{O}$. Each subset $U \in \mathcal{I}$ or $\mathcal{O}$ of factors defines a conjunctive component, represented by a one-hot code over its $n^{|U|}$ possible value combinations. For example, $\mathcal{I} = \{AB, CDE\}$ specifies the input in the recombination task (Eq. 8). Components are concatenated, and training includes all n^d combinations of actor values. A worked example illustrating the linear theory is in Fig. S5.

Any representation over these categorical factors can be decomposed into main effects of individual factors (e.g., A, B) and higher-order interactions (e.g., AB) via ANOVA (Appendix C.2). Intuitively, the effect of A captures variation associated with A that is shared across values of other factors, whereas an AB interaction captures how the effect of A changes with B. Let k_A and k_{AB} denote the strengths of these terms (formally these are the eigenvalues of the input/output/optimal hidden kernels). Marginal PS depends only on their *relative strength* q (Appendix C.2):

$$\overline{\mathrm{PS}}(A; B) = \frac{1 - q}{1 + (n - 1)q}, \qquad q = \frac{k_{AB}}{k_A} \in [0, 1]. \tag{9}$$

Thus $\overline{\mathrm{PS}}(A; B) = 1$ when the AB interaction vanishes ($q = 0$), and $\overline{\mathrm{PS}}(A; B) = 0$ when the interaction is as strong as the individual effect ($q = 1$).

For the optimal linear network, the relative strength of the hidden layer is simply the geometric mean of the relative strengths in the input and output (in the limit $\lambda \downarrow 0$, Appendix C.2):

$$q_h = \sqrt{q_{\mathrm{in}} q_{\mathrm{out}}}. \tag{10}$$

Since PS decreases monotonically with q, hidden PS lies between input and output PS. In particular, if the AB interaction is absent from either the input or output, then $q_h = 0$ and the hidden PS $\overline{\mathrm{PS}}_h(A; B) = 1$ (assuming A is present in both the input and output, so that $k_A^{(h)} > 0$).

The input and output strengths can be read directly from the task components. A one-hot component $[AB]$ contains an A effect, a B effect, and an AB interaction with equal strengths. More generally, a component contains all individual effects and high-order interactions formed from all its subsets: for example, $[ABC]$ contains A, B, C, AB, AC, BC, and ABC. Contributions from different components add, e.g., adding component $[AC]$ strengthens A without strengthening AB.

An implication of Eq. 10 is that **output-irrelevant factors are discarded entirely in the hidden layer**: every individual effect or interaction involving such a factor has zero output strength, and hence zero strength in the hidden layer. ReLU networks show the same behavior (Fig.S7a).

5.2 Linear theory explains how rebinding promotes abstraction

Unshared interactions yield full abstraction. If no input component contains both A and B, then $k_{AB}^{\mathrm{in}} = q_{\mathrm{in}} = 0$, so $q_h = \sqrt{q_{\mathrm{in}} q_{\mathrm{out}}} = 0$ and $\overline{\mathrm{PS}}_h(A; B) = 1$; the same holds if AB is absent from the output. Thus, a binding AB present on only one side of the input-output mapping is removed in the hidden representation, making A and B fully abstract with respect to each other. More strongly, if no interaction involving A is shared between the input and output, then A's coding is invariant across all combinations of the other factors, not only after marginalization (Appendix C.2).

This is the structure of both the rule transfer task and the recombination task: no higher-order interaction is shared between the input and output, so the linear theory predicts perfect abstraction. ReLU networks follow this prediction except for factor pairs bound at the output ($O_c R$ and AC, BD). The linear network cannot synthesize conjunctions absent in the input, but ReLU networks can construct them for the readout, thus leading to partial instead of perfect abstraction.

Reuse across partners increases abstraction. When AB is bound in both the input and output, the interaction is retained in the hidden layer and abstraction is incomplete. However, if A is also bound with other partners that exclude B (e.g., $[AC]$) in either input or output components, this

increases k_A without increasing k_{AB}, which lowers q_h and increases $\overline{\mathrm{PS}}_h(A;B)$. Thus, even when the original AB binding is retained, reusing A across distinct partners makes its representation more abstract with respect to B.

Figure 4 illustrates this graded effect. We fix the input at $[AB], [CD], [EF]$ and begin with output $[AB]$, for which $q_{\text{in}} = q_{\text{out}} = 1$ and therefore $\overline{\mathrm{PS}}_h(A;B) = 0$. We then successively add $[AC]$, $[AD]$, $[AE]$, and $[AF]$ to the output. Each new component strengthens A without strengthening AB, so the linear theory predicts increasing abstraction of A from B. ReLU networks show the same monotonic trend. Reuse in the input has similar effects (Fig. S6).

Together, these results formalize the rebinding mechanism: changing or diversifying a factor's partners either removes a particular interaction entirely or weakens it relative to the factor's main effect, both of which increase PS.

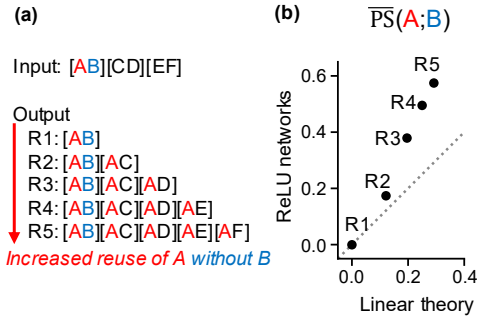


Figure 4: **Reuse across distinct partners promotes abstraction.** **(a)** Tasks R1–R5 share the same input and progressively add output bindings of A. **(b)** Linear predictions and ReLU simulations of $\overline{\mathrm{PS}}(A;B)$.

6 Linear predictions extend to nonlinear networks

6.1 Nonlinear networks broadly track linear theory predictions

We test linear predictions on a bank of tasks with $d = 6$ (Appendix B.5). We first test the prediction that unshared interactions yield full abstraction by training ReLU networks on 50 randomly selected tasks in which no higher-order interaction is shared between the input and output. To ensure that abstraction cannot arise from directly presented factors, these tasks contain no single-factor input or output components. Linear theory predicts perfect abstraction on these tasks; ReLU networks do not typically reach this limit, but their PS is substantially higher than a random-task baseline (Fig. 5a).

We then test if reuse across distinct partners increases abstraction on 200 random tasks. For each pair of output-relevant factors bound in both the input and output, define $R_{\text{reuse}}(A;B) = \log \frac{d_{\mathcal{I}}(A)d_{\mathcal{O}}(A)}{d_{\mathcal{I}}(AB)d_{\mathcal{O}}(AB)}$, where $d_{\mathcal{I}}(S)$ and $d_{\mathcal{O}}(S)$ are the numbers of input and output components containing S. This measures the reuse of A across input and output components relative to the conjunction AB. Across tasks, this reuse score is positively correlated with $\overline{\mathrm{PS}}(A;B)$ (Fig. 5b). More broadly, predicted PS values are positively correlated with those of nonlinear networks (Fig. S7b).

6.2 Depth separates abstraction and recombination

A single ReLU hidden layer must both extract reusable factors and construct conjunctions required by the readout. We next ask whether a second hidden layer allows these computations to separate across layers.

In the recombination task of Section 4, output-bound factor pairs are only partially abstract in a one-hidden-layer network. They become nearly fully abstract in the first layer of a two-hidden-layer network (Fig. 5c). Their PS decreases in the second hidden layer as the representation moves toward the conjunctive output. The same pattern extends across the task bank: adding a second hidden layer substantially increases abstraction in the first hidden layer (Fig. S8a). For tasks with no shared higher-order interactions, the first hidden layer achieves near-perfect abstraction as predicted by linear theory (Fig. S8b). These results suggest that depth allows the network to separate two steps: early layers extract reusable, abstract factor representations, while later layers recombine these factors into the conjunctions required by the output. Linear theory captures the first step especially well.

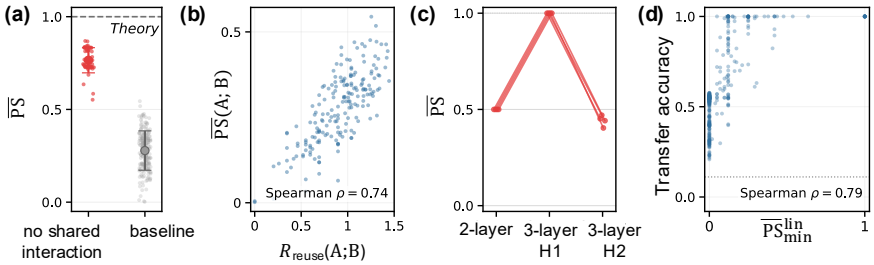


Figure 5: **Linear predictions in ReLU networks and transfer on task bank. (a)** Mean $\overline{\mathrm{PS}}$ across all output-relevant factor pairs in 50 tasks with no higher-order interactions shared between input and output. Baseline is across random tasks for factor pairs bound in both the input and output. Mean ± SD across tasks. **(b)** Reuse score versus marginal PS for factor pairs shared by input and output in 193 tasks eligible from the 200-task bank. **(c)** Recombination task $\overline{\mathrm{PS}}$ in 2-layer and 3-layer networks (H1 and H2), for output-bound factor pairs. **(d)** Mean accuracy on heldout contexts in the transfer task versus worst-pair PS predicted by linear theory. Each dot in **a–b** is one task averaged over three network seeds and factor pairs. Each dot in **(c)** is a factor pair and in **(d)** a transfer task.

7 Abstract representations support transfer to held-out contexts

Do these abstract representations facilitate downstream learning? We test this in a simple transfer setting. We freeze the hidden representation learned from the original task and train a new linear readout for a novel two-factor conjunction using data from only one context, defined by one combination of the remaining factor values, and then test it on held-out contexts (Appendix B.6). If the relevant factors are represented abstractly, a readout learned in one context should transfer to others, allowing the new conjunction to be learned from experience in only a single context. We therefore ask whether the abstraction predicted from the original task structure predicts transfer performance.

We expect transfer to depend on how abstract each factor in the new conjunction is across all other context factors. Because a single strongly context-dependent factor could limit transfer, we summarize the linear-theory prediction by the weakest pair:

$$\overline{\mathrm{PS}}^{\mathrm{lin}}_{\mathrm{min}} = \min_{\substack{T \in \mathcal{V}_{\mathrm{transfer}} \\ C \notin \mathcal{V}_{\mathrm{transfer}}}} \overline{\mathrm{PS}}_{\mathrm{lin}}(T; C), \tag{11}$$

where $\mathcal{V}_{\mathrm{transfer}}$ denotes factors in the new tested conjunction. Thus $\overline{\mathrm{PS}}^{\mathrm{lin}}_{\mathrm{min}}$ is the lowest predicted PS of any transfer-relevant factor across any context factor. Across tasks, transfer performance is strongly correlated with this metric (Fig. 5d).

8 Discussion

Our results show how compositional task structure can induce abstract factor representations from conjunctively encoded observations. We identify rebinding as a mechanism for abstraction: representation of a factor reused with different partners will be more abstract. In the rule transfer task, this abstract geometry is implemented by rectangular mixed-selective subpopulations in ReLU networks. In a more general task family, our linear theory predicts the degree of abstraction of each factor directly from the input–output structure. These predictions broadly extend to nonlinear networks, and the resulting abstraction tracks transfer to held-out contexts.

Our work leaves several directions open. Because training includes all factor combinations, we do not address generalization to unseen combinations or novel factor values. Our one-hot codes deliberately remove structure that could support such generalization. Extending the analysis to structured encodings would clarify how task-induced compositional structure interacts with structure already in the observations. Finally, our exact nonlinear theory is restricted to the rule transfer task, whereas the general theory is linear. Extending the nonlinear theory could reveal when linear predictions persist and how nonlinear networks construct the conjunctions required for readout.

AI USE STATEMENT

In this work, we used generative AI tools to help develop theoretical models, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design and provide feedback on research methodology and experiments, implement methods (including code that generates the task inputs and outputs as we defined), and interpret results. Assistance with translation, cleaning or reformatting datasets, and qualitative or thematic data analysis are not applicable to this work. We have reviewed all AI-assisted work. LLM-generated mathematical proofs were independently checked and LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Matteo Alleman, Jack W. Lindsey, and Stefano Fusi. Task structure and nonlinearity jointly determine learned representational geometry. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/713051bc96d98f3a8e35a27823bf2813-Abstract-Conference.html.

Silvia Bernardi, Marcus K. Benna, Mattia Rigotti, Jérôme Munuera, Stefano Fusi, and C. Daniel Salzman. The geometry of abstraction in the hippocampus and prefrontal cortex. *Cell*, 183(4):954–967.e21, 2020. doi: 10.1016/j.cell.2020.09.031. URL https://www.sciencedirect.com/science/article/pii/S0092867420312289.

Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11867. URL https://ojs.aaai.org/index.php/AAAI/article/view/11867.

Wenlin Chen, Julien Horwood, Juyeon Heo, and José Miguel Hernández-Lobato. Leveraging task structures for improved identifiability in neural network representations. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=WLcPrq6pu0.

Hristos S. Courellis, Juri Minxha, Araceli R. Cardenas, Daniel L. Kimmel, Chrystal M. Reed, Taufik A. Valiante, C. Daniel Salzman, Adam N. Mamelak, Stefano Fusi, and Ueli Rutishauser. Abstract representations emerge in human hippocampal neurons during inference. *Nature*, 632 (8026):841–849, 2024. doi: 10.1038/s41586-024-07799-x. URL https://www.nature.com/articles/s41586-024-07799-x.

Will Dorrell, Kyle Hsu, Luke Hollingsworth, Jin Hwa Lee, Jiajun Wu, Chelsea Finn, Peter E. Latham, Tim E. J. Behrens, and James C. R. Whittington. Range, not independence, drives modularity in biologically inspired representations. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/c07d71ff0bc042e4b9acd626a79597fa-Abstract-Conference.html.

Takuya Ito, Tim Klinger, Douglas H. Schultz, John D. Murray, Michael W. Cole, and Mattia Rigotti. Compositional generalization through abstract representations in human and artificial neural networks. In *Advances in Neural Information Processing Systems*, volume 35, pp. 32225–32239, 2022. doi: 10.52202/068431-2335. URL https://papers.nips.cc/paper_files/paper/2022/hash/d0241a0fb1fc9be477bdfde5e0da276a-Abstract-Conference.html.

Arthur Jacot, Eugene Golikov, Clément Hongler, and Franck Gabriel. Feature learning in l_2-regularized dnns: Attraction/repulsion and sparsity. *Advances in Neural Information Processing Systems*, 35:6763–6774, 2022.

Devon Jarvis, Richard Klein, Benjamin Rosman, and Andrew M Saxe. Compositionality and systematicity emerge from iterated learning in deep linear networks. *Proceedings of the National Academy of Sciences*, 123(19):e2509739123, 2026.

W. Jeffrey Johnston and Stefano Fusi. Abstract representations emerge naturally in neural networks trained to perform multiple tasks. *Nature Communications*, 14:1040, 2023. doi: 10.1038/s41467-023-36583-0. URL https://www.nature.com/articles/s41467-023-36583-0.

W Jeffrey Johnston and Stefano Fusi. Modular representations emerge in neural networks trained to perform context-dependent tasks. *bioRxiv*, 2024.

Samuel Lippl and Kimberly Stachenfeld. When does compositional structure yield compositional generalization? a kernel theory. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/572a6f16ec44f794fb3e0f8a310acbc6-Abstract-Conference.html.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4114–4124. PMLR, 2019. URL https://proceedings.mlr.press/v97/locatello19a.html.

Francesco Locatello, Ben Poole, Gunnar Rätsch, Bernhard Schölkopf, Olivier Bachem, and Michael Tschannen. Weakly-supervised disentanglement without compromises. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6348–6359. PMLR, 2020. URL https://proceedings.mlr.press/v119/locatello20a.html.

J. C. Raven. *Progressive Matrices: A Perceptual Test of Intelligence*. H. K. Lewis, London, 1938.

Reidar Riveland, Alexandre Pouget, and Peter Latham. Structural composition enables very fast learning. *bioRxiv*, 2026. doi: 10.64898/2026.07.14.738142.

Arda Sahiner, Tolga Ergen, John Pauly, and Mert Pilanci. Vector-output relu neural network problems are copositive programs: Convex analysis of two layer networks and polynomial-time algorithms. *arXiv preprint arXiv:2012.13329*, 2020.

Jiaqi Shang, Gabriel Kreiman, and Haim Sompolinsky. Unraveling the geometry of visual relational reasoning. *Scientific Reports*, 16:22146, 2026. doi: 10.1038/s41598-026-47123-3.

Nilesh Tripuraneni, Michael I. Jordan, and Chi Jin. On the theory of transfer learning: The importance of task diversity. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://papers.nips.cc/paper/2020/hash/59587bffec1c7846f3e34230141556ae-Abstract.html.

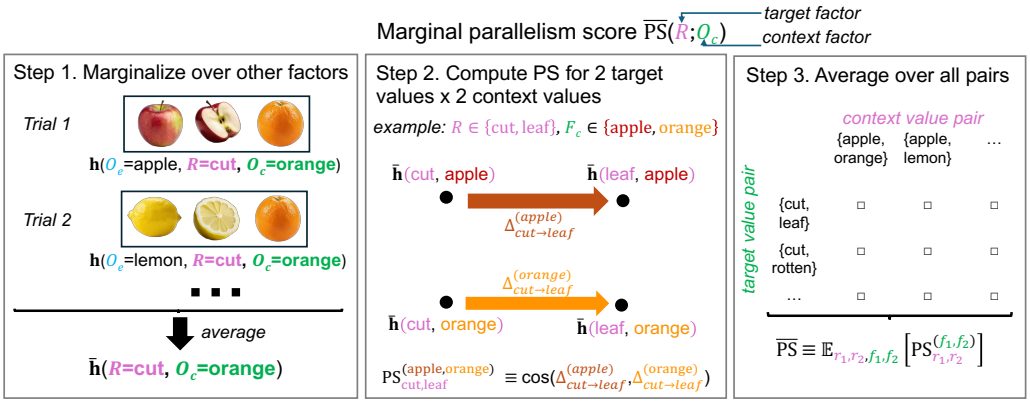
Pantelis Vafidis, Aman Bhargava, and Antonio Rangel. Disentangling representations through multi-task learning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/ab1b7c99395064743acbc6c694e4253a-Abstract-Conference.html.

Albert J. Wakhloo, Will Slatton, and SueYeon Chung. Neural population geometry and optimal coding of tasks with shared latent structure. *Nature Neuroscience*, 29(3):682–692, 2026. doi: 10.1038/s41593-025-02183-y.

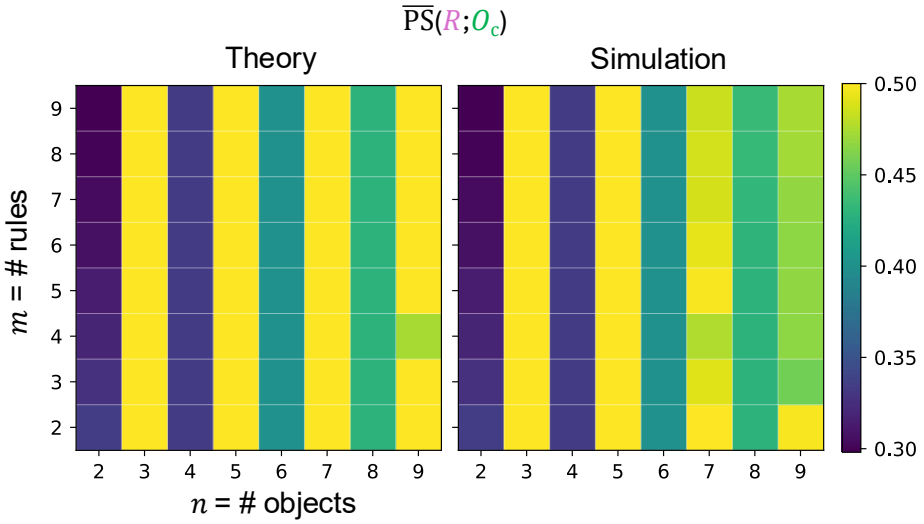
Bin Wang, W. Jeffrey Johnston, and Stefano Fusi. A mathematical theory for understanding when abstract representations emerge in neural networks. *arXiv preprint arXiv:2510.09816*, 2026. doi: 10.48550/arXiv.2510.09816. URL https://arxiv.org/abs/2510.09816v2. Version 2.

William W. Yang, Andrew M. Saxe, and Peter E. Latham. How are linear representations learned? exact solutions to the dynamics of abstraction. *arXiv preprint arXiv:2607.08843*, 2026. doi: 10.48550/arXiv.2607.08843. URL https://arxiv.org/abs/2607.08843.

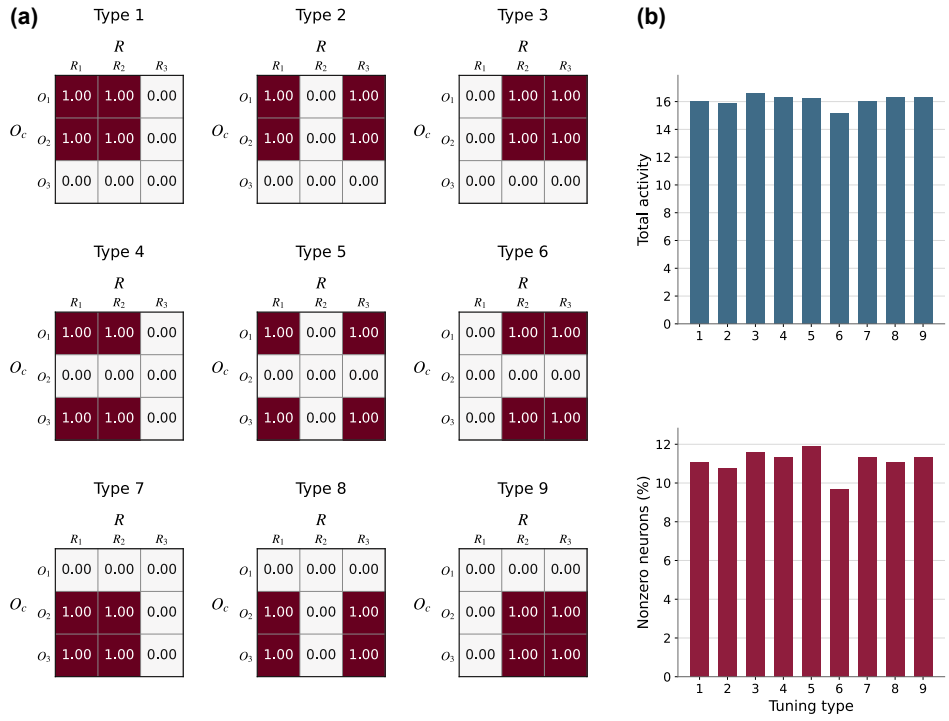
A SUPPLEMENTARY FIGURES



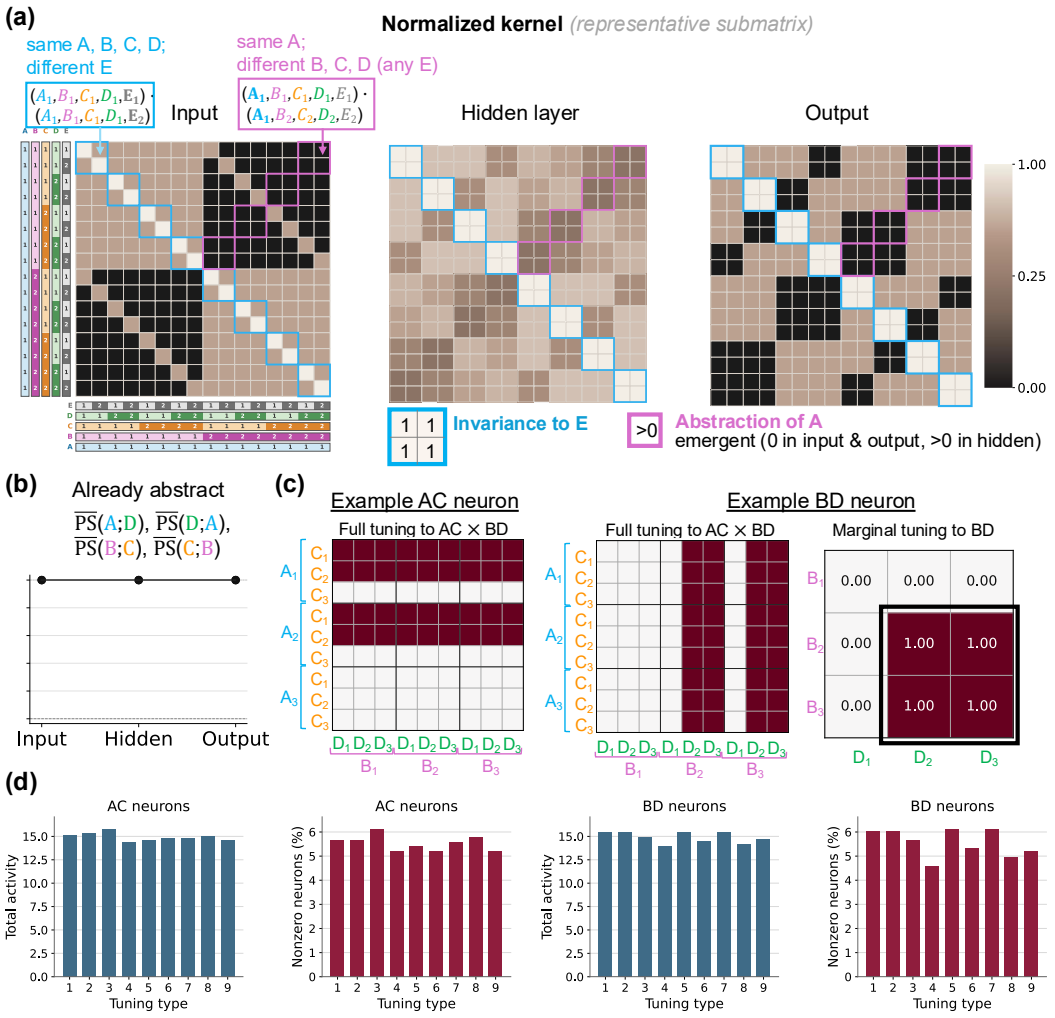
Supplementary Figure S1: **Illustration of the marginal parallelism score** for $\overline{\mathrm{PS}}(R;O_c)$ in the rule transfer task.



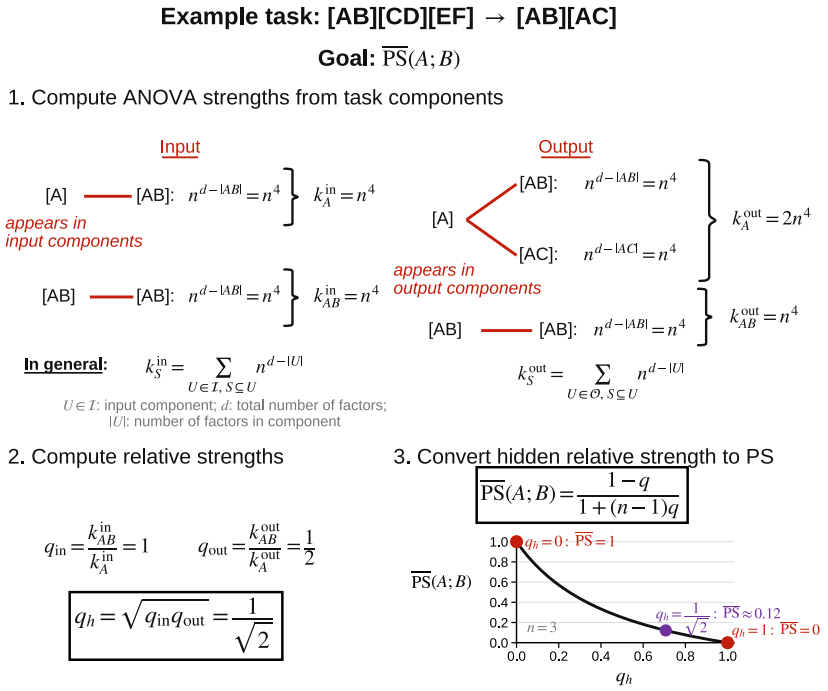
Supplementary Figure S2: **Theory and simulations of marginal rule PS in the rule transfer task across task sizes.**



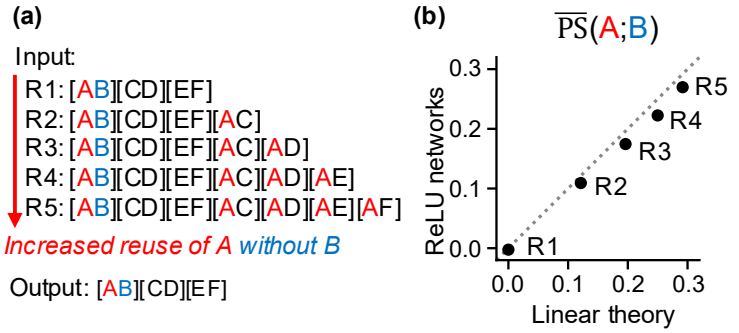
Supplementary Figure S3: **All nonzero neuron subpopulations in example trained ReLU network on the rule transfer task**. $m = n = 3$. **(a)** Marginal tuning over R and O_c (averaging over O_e) for example neurons from each specific type of rectangular tuning. Each heatmap is scaled by the neuron's maximum activity. All nonzero neurons (at threshold $\epsilon = 10^{-3}$) belong in one of these types. **(b)** Distribution across neuron types. Top: total unnormalized marginal activity summed across all neurons and the nine $O_c \times R$ marginal conditions. Bottom: percentage of all nonzero neurons belonging to each type. See Appendix B.2 for neuron type classification.



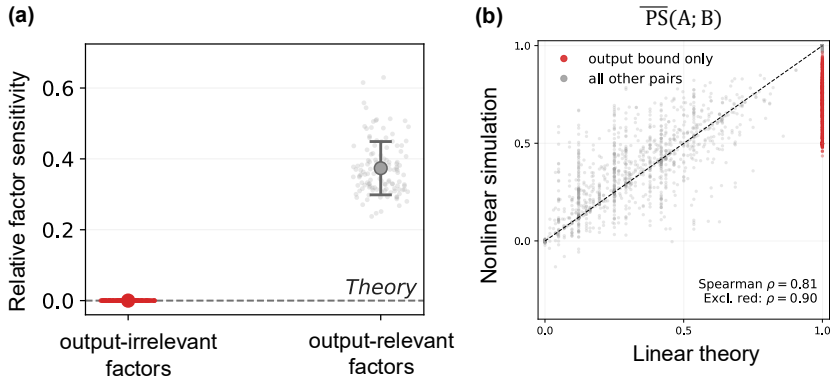
Supplementary Figure S4: **ReLU network simulations on the recombination task**. **(a)** Representative normalized kernel submatrices for $n = 2$ (see Fig. 2b). We show the submatrix for trials with $A = A_1$. Trials are ordered first by A, then B, C, D, and E. **(b)-(d)** Simulations from $n = 3$. **(b)** Marginal PS at the input, hidden layer, and output for pairs of output-relevant factors whose conjunctions are not in either the input or output. **(c)** Tunings of example AC- and BD-selective neurons. Left: tuning to A, B, C, D values (averaged over E) for example AC-selective neuron. Middle and right: tuning of example B, D-selective neuron. **(d)** Distribution across neuron types for AC- and BD-selective neurons (see Fig. S3; neuron types are defined the same way as Fig. S3a), see method details in Appendix B.3. Percentage bars show the fraction of all active neurons.



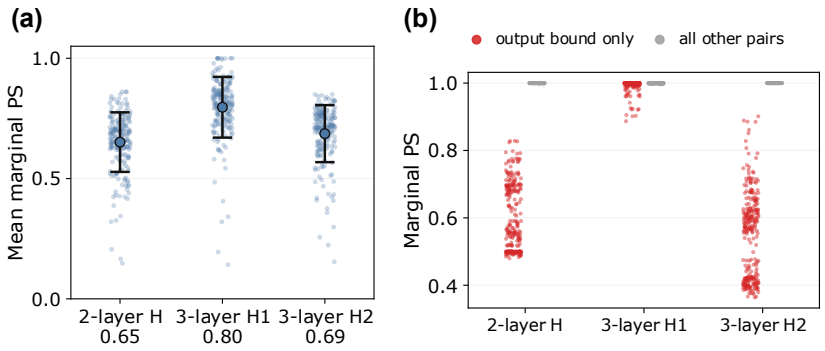
Supplementary Figure S5: **Illustration of the linear theory prediction from task structure.** For the example task $[AB][CD][EF] \rightarrow [AB][AC]$ and target abstraction $\overline{\mathrm{PS}}(A;B)$, we compute the ANOVA strengths k_A and k_{AB} directly from the input and output components, convert them to relative interaction strengths $q = k_{AB}/k_A$, and obtain the hidden-layer value $q_h = \sqrt{q_{\mathrm{in}}q_{\mathrm{out}}}$. The marginal parallelism score then follows from $\overline{\mathrm{PS}}(A;B) = \frac{1-q}{1+(n-1)q}$; for $n = 3$, $q_h = 1/\sqrt{2}$ gives $\overline{\mathrm{PS}}_h(A;B) \approx 0.12$.



Supplementary Figure S6: **Reuse across distinct partners in the input also promotes abstraction.** **(a)** Tasks R1–R5 share the same output and progressively add input bindings of A. **(b)** Linear predictions and ReLU simulations of $\overline{\mathrm{PS}}(A;B)$.



Supplementary Figure S7: **Nonlinear networks broadly follow linear theory predictions for factor relevance and abstraction.** Both panels use one-hidden-layer ReLU networks trained on the random bank of 200 tasks. **a** Relative factor sensitivity for output-irrelevant factors compared to output-relevant factors. Each dot is one task, averaged across all related factors and network seeds. Marker and errorbars are mean ± SD across tasks. **b** Comparison of linear theory predictions and nonlinear network simulations for marginal PS. Each dot is the marginal PS for a single factor pair on a single task; for simulations, we average across three network seeds. We include all pairs of output-relevant factors in each task. Red dots are for factor pairs bound in the output but not in the input; grey dots are all other factor pairs. We compute Spearman correlation across all dots, and across grey dots only excluding red dots.



Supplementary Figure S8: **Effect of depth on abstraction**. (a) Marginal PS on 200 random tasks. Each dot is a single task: we average the marginal PS across all output-relevant factor pairs and three network seeds. Markers and error bars indicate the mean (also shown in text) ± SD. (b) Marginal PS on 50 tasks with no shared higher-order interactions between the input and output (same as Fig. 5a). Each dot is an ordered pair of output-relevant factors; red dots are factor pairs bound in the output only.

B SIMULATION METHODS

B.1 COMMON TRAINING CONVENTIONS

Network weights and biases are initialized by PyTorch default initialization multiplied by 10^{-3} to place training in the feature learning regime. Networks are trained using full-batch Adam on all possible trials, using the same loss as the theoretical setup with regularization strength $\lambda = 10^{-2}$ (Eq. 3; both weights and biases are regularized). Learning rates use a cosine decay schedule with an extended final phase: if η_0 and $\eta_{\min}$ are the initial and final learning rates and T_{decay} is the decay horizon, then at update t

$$\eta_t = \eta_{\min} + \frac{\eta_0 - \eta_{\min}}{2}\left[1 + \cos\left(\pi \frac{\min(t, T_{\text{decay}})}{T_{\text{decay}}}\right)\right]. \tag{12}$$

After T_{decay}, the learning rate is fixed at $\eta_{\min}$. We use $\eta_0 = 10^{-3}$ and $\eta_{\min} = 10^{-5}$. All reported networks achieve perfect training accuracy, defined as the fraction of trials for which the argmax prediction is correct in every output component's one-hot block.

B.2 RULE TRANSFER TASK

Architecture and training We train one-hidden-layer networks with width 4096. Networks are trained for $300{,}000$ full-batch updates with $T_{\text{decay}} = 200{,}000$. We train two networks for each combination of $m, n \in \{2, \ldots, 9\}$; Supplementary Figure S2 reports the average across the two networks.

Single neuron analysis We use one example network with $m = n = 3$. We average activity over O_e to obtain the 3×3 marginal tuning matrix to $O_c \times R$. We label a matrix cell as nonzero if its value is above 10^{-3}; a neuron has nonzero activity if at least one cell is nonzero. We empirically find that all nonzero neurons have exactly four nonzero cells that correspond to the combinations of two cue object values and two rules (they form rectangular tuning). All $\binom{3}{2}^2 = 9$ possible rectangular tunings (i.e. collection of nonzero cells) exist in the network, which we label as tuning types.

B.3 RECOMBINATION TASK

Architecture and training For $n = 2$ and $n = 3$, we each train an example one-hidden-layer network with width 4096 for $120{,}000$ full-batch updates with $T_{\text{decay}} = 80{,}000$. To test the effect of depth, for $n = 3$ we train a two-hidden-layer ReLU network with width 512 for both layers using the same objective and optimization schedule.

Single neuron analysis We use the example one-hidden-layer network with $n = 3$. AC- and BD-selective neurons are classified by comparing each neuron's trial-by-trial activity with its AC- and BD-marginal activity. For example, the AC-marginal activity is

$$\mu_j^{AC}(a, c) = \mathbb{E}_{B,D,E}[h_j(A, B, C, D, E) \mid A = a, C = c].$$

We compute the relative squared residual as

$$r_j^{AC} = \frac{\sum_z [h_j(z) - \mu_j^{AC}(A(z), C(z))]^2}{\sum_z h_j(z)^2},$$

and define r_j^{BD} analogously. We consider only nonzero active neurons, defined as those whose maximum activity across all trials exceeds 10^{-3}. We classify a neuron as AC-selective if its residual is no greater than 10^{-5}, and similarly for BD. All active neurons in the example network pass this residual threshold for exactly one class. Within each class, we use the same method as in the rule transfer task to classify tuning type.

B.4 EXAMPLES OF REUSE

The task inputs and outputs are shown in Fig. 4a and Supplementary Fig. S6a. Both examples contain five tasks involving six factors $A, \ldots, F$, each with $n = 3$ values. In the output reuse example, the

input is held fixed while the output progressively adds bindings of A with distinct factors. Similarly, in the input reuse example, the output is held fixed while the input progressively adds bindings of A. For each task, we train a one-hidden-layer ReLU network with hidden width 512 for $80{,}000$ full-batch updates, with $T_{\text{decay}} = 40{,}000$.

B.5 General task bank

Task generation. The bank uses $d = 6$ factors, each with $n = 3$ values. Input and output each contain 2–5 components with equal probability. Each component is sampled independently and contains 1–4 factors, with probabilities 0.10, 0.40, 0.40, and 0.10, respectively; we chose this distribution to reduce the presence of single factors (which makes abstraction easy) and four-factor components (which increase the dimensionality significantly). For each target number of factors, all factor subsets are sampled uniformly randomly. Repeated components within the input (or within the output) are resampled so all components are unique (note a component can appear in both input and output), and duplicate tasks are resampled. We only keep tasks where the input includes all six factors; this guarantees that all output factors are included in the input and hence the task is deterministic. In total, we generate 10,000 valid task candidates. From this large set of candidates, we randomly select 200 tasks; this forms the standard bank that we use for testing. In the remaining candidate tasks, we additionally identify all tasks where no higher-order interaction is shared between the input and output, no input or output component is a single factor, and there are at least two output-relevant factors. We randomly select 50 tasks from this subset.

Network training. For every task we trained three one-hidden-layer networks and three two-hidden-layer networks. All hidden layers have width 512 and use ReLU. We train for $80{,}000$ full-batch updates with $T_{\text{decay}} = 40{,}000$. Three two-hidden-layer networks did not achieve perfect training accuracy and additional models with alternative seeds were trained to replace them. Unless stated otherwise, the main task bank analyses use the representation from the one-hidden-layer networks (Fig. 5a,b,d and Fig. S7). The depth analysis in Fig. 5c and Fig. S8 compares one-hidden-layer networks with two-hidden-layer networks.

Comparison with linear predictions For the 50 tasks with no shared higher-order interactions, we compute marginal PS for every ordered pair of output-relevant factors; we average over all such pairs and over three networks for each task. For the baseline, we use the 200 random tasks, and for each task average over pairs of output-relevant factors that are bound in both the input and the output (so the linear theory predicts marginal PS below 1); this gives 1,460 eligible factor pairs across 193 tasks. These results are reported in Fig. 5a. For testing reuse in Fig. 5b, we use the same set of factor pairs (i.e. all output-relevant factor pairs that are bound in both the input and output in the 200 random tasks), and for each pair compute the marginal PS and reuse score $R_{\text{reuse}}(A; B)$ (as defined in the main text).

For testing if output-irrelevant factors are not represented in the hidden layer, we compute the relative factor sensitivity: for factor i, let $\boldsymbol{\mu}_i(\mathbf{z}_{-i}) = \frac{1}{n}\sum_{a=1}^{n} \mathbf{h}(a, \mathbf{z}_{-i})$ be the mean hidden activity across its values while holding all other factors fixed. We define its relative sensitivity as

$$S_i = \frac{\mathbb{E}_{\mathbf{z}_{-i}}\left[\frac{1}{n}\sum_{a=1}^{n} \|\mathbf{h}(a, \mathbf{z}_{-i}) - \boldsymbol{\mu}_i(\mathbf{z}_{-i})\|_2^2\right]}{\mathbb{E}_{\mathbf{z}}\left[\left\|\mathbf{h}(\mathbf{z}) - \overline{\mathbf{h}}\right\|_2^2\right]},$$

where $\overline{\mathbf{h}}$ is the mean hidden activity across all trials. Thus, S_i is the variance in hidden activity induced by factor i, normalized by the total hidden activity variance.

B.6 Transfer task

We take single-hidden-layer networks trained on the random 200-task bank. For each task, we identify all unordered pairs of output-relevant factors that are neither bound in the input nor the output. We restrict to tasks with at least four output-relevant factors, so the context of the transfer task contains at least two output-relevant factors. This gives 142 original tasks and 328 transfer tasks, where a transfer task is a unique combination of an original task and a target factor pair.

For each transfer task, we take every network trained on the original task, freeze their hidden representation, and train a separate linear softmax classifier (logits $g(H) = WH + b$, followed by softmax) with nine output classes for the target factor pair conjunction. We define a context as a unique combination of values for all four factors outside the transfer target pair (this gives $3^4 = 81$ contexts). For training, we only use the nine trials from a single random context (covering all possible combinations of the transfer target pair); we repeat for 3 different context splits for a given transfer task. We use the cross-entropy loss (averaged across training trials) with L_2 regularization on the weights (excluding the bias), using regularization strength 10^{-4}. Before probe fitting, we divide the hidden activity by its root-mean-square value computed across all 729 trials and all hidden units for normalization. We initialize weights and bias to zero and use full-batch Adam for 1,000 updates, with a cosine learning-rate decay from 3×10^{-2} to 3×10^{-4} with $T_{\text{decay}} = 1{,}000$. All fitted probes achieved training accuracy 1.0. We then test each probe on all other contexts. We average the test accuracy across all heldout-contexts, context splits, and the three trained networks.

C THEORETICAL RESULTS

In this Appendix, we present our main theoretical results regarding the optimal representation in ReLU and linear networks for the tasks studied in the main text. For readability, we will state the main results for ReLU network here and present the detailed proof in the supplementary material. The results for linear network will be presented in a self-contained manner.

C.1 OPTIMAL REPRESENTATION IN TWO-LAYER RELU NETWORK FOR THE RULE TRANSFER TASK

We summarize here the characterization of the global optimum used in the corresponding section of the main text. Detailed proofs for these results are provided in the Supplementary Material and the main idea is to use the measure-space optimization framework developed in Wang et al. (2026) for characterizing the optimal hidden kernel and single-neuron tuning. We use the notation of the main text: n is the number of objects, m is the number of rules, and trials are indexed by $(O_e, R, O_c) \in \mathcal{O} \times \mathcal{R} \times \mathcal{O}$. The total number of input vectors (trial conditions) is $n \times m \times n$ and the input/output kernel can be viewed as a matrix in the tensor product space $\mathbb{R}^n \otimes \mathbb{R}^m \otimes \mathbb{R}^n$.

C.1.1 THE OPTIMAL HIDDEN KERNEL

Let

$$P_O = \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top, \qquad Q_O = I_n - P_O, \qquad P_R = \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top, \qquad Q_R = I_m - P_R, \tag{13}$$

and let $P_E = \mathbf{1}_n\mathbf{1}_n^\top/n$ project onto the constant example-object direction. The optimal hidden kernel is unique and has only four nonzero terms:

$$\boxed{K_H^* = P_E \otimes (\rho_0 P_R \otimes P_O + \rho_O Q_R \otimes P_O + \rho_R P_R \otimes Q_O + \rho_{OR} Q_R \otimes Q_O)\,.} \tag{14}$$

Thus every global optimum is constant along the O_e direction, consistent with the invariance to example object described in the main text (Eq. 5). The four terms in Eq. 14 have eigenvectors that are mutually orthogonal to each other and thus the coefficients are the corresponding eigenvalues in the mean, cue-object, rule, and cue-object–rule interaction subspaces, respectively.

A convenient way to solve these eigenvalues is as follows. It turns out that the optimal single-neuron tuning to cue objects and rules has rectangle shapes (Appendix C.1.2). A rectangle type is specified by its sizes (k, ℓ), where $1 \le k \le n$ cue objects and $1 \le \ell \le m$ rules are selected. Define

$$p_O = \frac{k^2}{n}, \qquad q_O = \frac{k(n-k)}{n(n-1)}, \tag{15}$$

$$p_R = \frac{\ell^2}{m}, \qquad q_R = \frac{\ell(m-\ell)}{m(m-1)}, \tag{16}$$

and

$$\mathbf{v}_{k\ell} = n\,(p_O p_R,\ q_O p_R,\ p_O q_R,\ q_O q_R)^\top\,. \tag{17}$$

If $\nu_{k\ell} \geq 0$ is the total squared-amplitude mass assigned to all rectangles of size (k, ℓ), then

$$(\rho_0, \rho_O, \rho_R, \rho_{OR})^\top = \sum_{k=1}^{n} \sum_{\ell=1}^{m} \nu_{k\ell} \mathbf{v}_{k\ell}. \tag{18}$$

For the input convention in the rule transfer encoding in the main text, put

$$x = \frac{k}{n}, \qquad y = \frac{\ell}{m}, \tag{19}$$

and define the normalized input cost

$$a_{k\ell} = \frac{m(x + y - 1)^2}{mn + 2m + 1} + x(1 - x) + my(1 - y). \tag{20}$$

The original regularized loss minimization problem (Eq.3) reduces to the convex program

$$\min_{\nu_{k\ell} \geq 0} \left\{ \sum_{k,\ell} a_{k\ell} \nu_{k\ell} + \frac{1}{\lambda + mn^2 + \rho_0} + \frac{n - 1}{\lambda + \rho_O} + \frac{m - 1}{\lambda + \rho_R} + \frac{(n - 1)(m - 1)}{\lambda + \rho_{OR}} \right\}, \tag{21}$$

with the ρ's given by Eq. 18. Therefore solving this finite program and inserting the resulting eigenvalues into Eq. 14 gives the globally optimal hidden kernel. There are typically multiple sets of $\{\nu_{k\ell}\}$ that all achieve the global minima of Eq. 21 but they all correspond to a unique set of $(\rho_0, \rho_O, \rho_R, \rho_{OR})$. The small-$\lambda$ results used in the main text are obtained by taking $\lambda \downarrow 0$.

Eq. 14 makes the rule PS used in the main text explicit.The marginal PS can be obtained from the optimal hidden kernel

$$\overline{\mathrm{PS}}(R; O_c) = \frac{\rho_R - \rho_{OR}}{\rho_R + (n - 1)\rho_{OR}}, \tag{22}$$

whenever the denominator is nonzero. Thus, the theoretical PS curves in the main-text theory panel can be computed directly from the unique optimal kernel. Moreover, it can be also be shown that $\overline{\mathrm{PS}}(R; O_c) \geq 2 - \sqrt{3}$.

C.1.2 Single-neuron tuning in the small-regularization regime

We can obtain the optimal single-neuron tuning in the small-λ limit. Up to a positive scaling and permutations of object and rule labels, every nonzero neuron in a global optimum has preactivation

$$a_{G,S}(O_e, R, O_c) = d\left(\mathbf{1}\{O_c \in G\} + \mathbf{1}\{R \in S\} - 1\right), \qquad d > 0, \tag{23}$$

for some nonempty $G \subseteq \mathcal{O}$ and $S \subseteq \mathcal{R}$. The post-activation after ReLU is therefore

$$\boxed{h_{G,S}(O_e, R, O_c) = d\,\mathbf{1}\{O_c \in G\}\mathbf{1}\{R \in S\}.} \tag{24}$$

Hence every neuron's activation is invariant across example objects O_e and has exactly the rectangular mixed selectivity described in the main text.

Parallelism as a function of rectangle size. First consider a single permutation-symmetric rectangle-size orbit (k, ℓ) (i.e. all neurons with rectangular tunings of size k and l, with equal weights across all possible choices of $|G|$ and $|S|$ with $k = |G|$ and $\ell = |S|$). Its contribution to the four components of the kernel is proportional to the orbit vector $\mathbf{v}_{k\ell}$ in Eq. 17.

If this is the only active rectangle size and it does not have pure-rule tuning $\ell < m$, then $\rho_R/\rho_{OR} = p_O/q_O = k(n - 1)/(n - k)$. Substituting into Eq. 22 gives

$$\boxed{\overline{\mathrm{PS}}_{k\ell}(R; O_c) = \frac{k - 1}{n - 1}.} \tag{25}$$

Thus the rule PS of a single rectangle-size orbit depends only on its cue-object width k, not on its rule width ℓ. By symmetry, if it does not have pure-cue tuning ($k < n$),

$$\boxed{\overline{\mathrm{PS}}_{k\ell}(O_c; R) = \frac{\ell - 1}{m - 1}.} \tag{26}$$

When multiple rectangle sizes are active and have positive mass, the total PS is a weighted average of these single-size values. Using the permutation-symmetric masses $\nu_{k\ell}$ from Eq. 18, define rule-tuning weight

$$D_{k\ell} = \frac{nk\ell(m-\ell)}{m(m-1)}. \tag{27}$$

Then

$$\boxed{\overline{\mathrm{PS}}(R; O_c) = \frac{\sum_{k,\ell} \nu_{k\ell} D_{k\ell} \dfrac{k-1}{n-1}}{\sum_{k,\ell} \nu_{k\ell} D_{k\ell}}}. \tag{28}$$

Similarly, defining

$$\widetilde{D}_{k\ell} = \frac{k(n-k)\ell}{n-1}, \tag{29}$$

the cue-object PS is

$$\boxed{\overline{\mathrm{PS}}(O_c; R) = \frac{\sum_{k,\ell} \nu_{k\ell} \widetilde{D}_{k\ell} \dfrac{\ell-1}{m-1}}{\sum_{k,\ell} \nu_{k\ell} \widetilde{D}_{k\ell}}}. \tag{30}$$

These weighted-average expressions depend only on the unique optimal kernel, even though its neuron-level realization need not be unique.

We also determine which rectangle cardinalities $(|G|, |S|) = (k, \ell)$ can carry positive mass in the supplementary materials. The complete small-λ map for the permitted rectangle sizes is summarized below. We write $k_\pm = (n \pm 1)/2$. For an odd m, let $r = (m+1)/2$, and for an even m, let $r = m/2$.

Boundary dimensions.

regime	positive-mass rectangle sizes (k, ℓ)
$n = m = 2$	$(1,1), (1,2), (2,1)$
$n \geq 3,\ m = 2$	$(1,2), (\lceil n/2 \rceil, 1)$
$n = 2,\ m \geq 3$ odd	$(1, (m+1)/2), (2,1)$
$n = 2,\ 4 \leq m \leq 10$ even	$(1, m/2), (1, m/2+1), (2,1)$
$n = 2,\ m \geq 12$ even	$(1, m/2), (2,1)$.

(31)

These low-dimensional cases are the only ones in which cue-only ($\ell = m$) or rule-only ($k = n$) neurons are optimal. In particular, for $m, n \geq 3$ as shown below, every nonzero neuron is mixed-selective to both cue object and rule.

Even number of objects ($n \geq 4$, $m \geq 3$). If m is odd, the unique active size is

$$(k, \ell) = \left(\frac{n}{2}, \frac{m+1}{2} \right). \tag{32}$$

If m is even, the active support is either the central singleton $(n/2, m/2)$ or the adjacent pair

$$\left(\frac{n}{2}, \frac{m}{2} \right), \qquad \left(\frac{n}{2}, \frac{m}{2} + 1 \right). \tag{33}$$

The adjacent pair occurs when $m \leq L(n)$, where

$$L(n) = \begin{cases} 18, & n = 4, \\ 32, & n = 6, \\ 42, & n = 8, \\ 5n+4, & 10 \leq n \leq 18, \\ 5n+6, & n \geq 20, \end{cases} \tag{34}$$

and the singleton occurs for $m > L(n)$.

Odd number of objects and odd number of rules ($n, m \geq 3$). For $n < m$, define

$$M(n) = \begin{cases} n^2 + 2n - 4, & n \in \{3, 5, 7\}, \\ n^2 + 2n - 2, & n \geq 9. \end{cases} \tag{35}$$

If $m \leq M(n)$, the active size is the central singleton (k_+, r). For $m \geq M(n) + 2$, the active sizes are either one rule width (k_+, j) or two adjacent widths $(k_+, j), (k_+, j + 1)$. The exact integer j is selected by the one-dimensional condition that can be found in the supplementary material: it is close to, but need not equal, the nearest half-rule width.

For $n \geq m$, all active neurons lie on the central rule column $\ell = r$. Define

$$A(m) = m^2 + m - 1, \qquad B(m) = \begin{cases} 17, & m = 3, \\ m^2 + m + 7, & m \geq 5. \end{cases} \tag{36}$$

Then the support is

$$\begin{cases} \{(k_+, r)\}, & n \leq A(m) - 2, \\ \{(k_-, r), (k_+, r)\}, & A(m) \leq n \leq B(m) - 2, \\ \{(k_-, r)\}, & n \geq B(m). \end{cases} \tag{37}$$

These cases are exhaustive on the odd integer lattice (m, n).

Odd number of objects and even number of rules ($n \geq 3, m \geq 4$). For $n < m$, the active sizes are either a singleton (k_+, j) or adjacent pairs $(k_+, j), (k_+, j+1)$, where the rule width j is selected by the same one-dimensional condition as in the case with odd number of objects and odd number of rules.

For $n \geq m$, define

$$N_1(m) = \begin{cases} (m^2 + m - 2)/2, & m \equiv 0 \pmod 4, \\ (m^2 + m - 4)/2, & m \equiv 2 \pmod 4, \end{cases} \tag{38}$$

and $N_{D,1}(m) = N_1(m) + \Delta_1(m)$, where

$$\Delta_1(m) = \begin{cases} 2, & m = 4, \\ 4, & m \equiv 0 \pmod 4,\ m \geq 8, \\ 4, & m \in \{6, 10, 14\}, \\ 6, & m \equiv 2 \pmod 4,\ m \geq 18. \end{cases} \tag{39}$$

The support then passes through three simple faces:

$$\begin{array}{c|l} m \leq n < N_1(m) & (k_+, r), (k_+, r + 1) \\ N_1(m) \leq n < N_{D,1}(m) & (k_+, r), (k_+, r + 1), (k_-, r + 1) \\ n \geq N_{D,1}(m) & (k_+, r), (k_-, r + 1). \end{array} \tag{40}$$

For any support listed above, the positive masses $\nu_{k\ell}$ are obtained by solving the active stationarity equations of Eq. 21. In general, although the optimal kernel is unique, the optimal collections of positive masses $\nu_{k\ell}$ that achieve this optimum are non-unique.

C.2 Optimal representation in two-layer linear network for general rebinding tasks

We now derive the linear-network result used in Section 5 assuming the hidden layer is wide enough. We first solve the explicitly regularized problem, which gives Eq. 9-10 in the small regularization limit $\lambda \downarrow 0$. This solution for hidden representation is the same as gradient descent from infinitesimal initialization on the unregularized problem. We will also present this equivalence at the end.

Task kernels and the common ANOVA basis. Let the task contain d categorical factors, each taking n values, and let all $P = n^d$ factor combinations be included in the training set. A component is indexed by a subset $U \subseteq [d]$ of factors and assigns a separate one-hot code to every joint value of the factors in U. If X_U denotes the component matrix, with trials as columns, then its kernel matrix is

$$(K_U)_{\mathbf{z},\mathbf{z}'} = \mathbf{1}\{\mathbf{z}_U = \mathbf{z}'_U\}, \tag{41}$$

where $\mathbf{z} = (z_1, \ldots, z_d) \in [n]^d$ indexes a trial and $\mathbf{z}_U = (z_i)_{i \in U}$ contains those components belonging to U.

For each factor define the projectors

$$P_i = \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top, \qquad Q_i = I_n - P_i. \tag{42}$$

For each subset $S \subseteq [d]$, define the ANOVA-sector projector

$$\Pi_S = \bigotimes_{i=1}^{d} R_i^{(S)}, \qquad R_i^{(S)} = \begin{cases} Q_i, & i \in S, \\ P_i, & i \notin S. \end{cases} \tag{43}$$

The projectors $\{\Pi_S\}_{S \subseteq [d]}$ are mutually orthogonal and sum to the identity on the trial space. Likewise define

$$M_i^{(U)} = \begin{cases} I_n, & i \in U, \\ nP_i, & i \notin U. \end{cases} \tag{44}$$

Since $I_n = P_i + Q_i$ and $\mathbf{1}_n\mathbf{1}_n^\top = nP_i$, Eq. 41 gives

$$K_U = \bigotimes_{i=1}^{d} M_i^{(U)} = n^{d-|U|} \sum_{S \subseteq U} \Pi_S. \tag{45}$$

Thus a conjunctive component U contains exactly projectors of every subset $S \subseteq U$, with its eigenvalue equal to $n^{d-|U|}$.

If the input concatenates the components in $\mathcal{I}$, then $K_X = X^\top X = \sum_{U \in \mathcal{I}} K_U$. Hence

$$K_X = \sum_{S \subseteq [d]} k_S^{\text{in}} \Pi_S, \qquad k_S^{\text{in}} = \sum_{\substack{U \in \mathcal{I} \\ S \subseteq U}} n^{d-|U|}. \tag{46}$$

Similarly,

$$K_Y = Y^\top Y = \sum_{S \subseteq [d]} k_S^{\text{out}} \Pi_S, \qquad k_S^{\text{out}} = \sum_{\substack{U \in \mathcal{O} \\ S \subseteq U}} n^{d-|U|}. \tag{47}$$

Therefore K_X and K_Y commute and share the ANOVA eigenbasis. The projector corresponding to $S = \varnothing$ is the constant direction. Hidden and output biases only affect this component, whereas all abstraction results below concern other components. We therefore focus on the weights on $S \neq \varnothing$. The above equations also show that $k_A^{\text{in}} \geq k_{AB}^{in}$.

More generally, the argument below only requires that K_X and K_Y commute. Choose a simultaneous orthonormal eigenbasis $V = [v_1, \ldots, v_P]$ and write

$$K_X = V \operatorname{diag}(s_1^2, \ldots, s_P^2) V^\top, \qquad K_Y = V \operatorname{diag}(t_1^2, \ldots, t_P^2) V^\top, \tag{48}$$

with $s_i, t_i \geq 0$. Whenever $s_i > 0$, define $u_i^X = Xv_i/s_i$, and whenever $t_i > 0$, define $u_i^Y = Yv_i/t_i$. Within their supports, the u_i^X and u_i^Y form orthonormal families and as we will show later, these are the input/output singular vectors.

Regularized MSE. Consider a linear hidden layer

$$H = W_1 X, \qquad \widehat{Y} = W_2 H, \tag{49}$$

and, on the nonconstant components, the same regularized squared-error objective as Eq. 3:

$$\mathcal{L}_\lambda(W_1, W_2) = \|Y - W_2 W_1 X\|_F^2 + \lambda \left(\|W_1\|_F^2 + \|W_2\|_F^2 \right). \tag{50}$$

For a fixed end-to-end map $A = W_2 W_1$, the minimum layerwise penalty is

$$\min_{W_2 W_1 = A} \left(\|W_1\|_F^2 + \|W_2\|_F^2 \right) = 2\|A\|_*, \tag{51}$$

where $\|A\|_*$ is the nuclear norm. Equality is attained by the balanced square-root factorization of the singular value decomposition (SVD) of A. Using this, Eq. 50 is equivalent to

$$\min_A \ \|Y - AX\|_F^2 + 2\lambda \|A\|_*. \tag{52}$$

Because the input and output sample kernels share the basis in Eq. 48, the optimum does not mix different sample modes. On a mode with $s_i, t_i > 0$, write

$$Au_i^X = a_i u_i^Y, \qquad a_i \geq 0. \tag{53}$$

By rotating the terms in Eq. 52 to this basis, we get

$$(t_i - a_i s_i)^2 + 2\lambda a_i. \tag{54}$$

Minimizing over a_i gives

$$a_i^* = \frac{[s_i t_i - \lambda]_+}{s_i^2}, \qquad [x]_+ := \max(x, 0). \tag{55}$$

Modes with $s_i = 0$ cannot be produced from the input and have $a_i^* = 0$.

Therefore, the optimal A matrix is $A = U_Y \mathrm{diag}(a_i^*) U_X^T = W_2^* W_1^*$, where U_X and U_Y collect the corresponding u_i^X and u_i^Y. We can always write

$$W_1^* = Q\,\mathrm{diag}(\sqrt{a_i^*})\,(U_X)^\top, \qquad W_2^* = U_Y\,\mathrm{diag}(\sqrt{a_i^*})\,Q^\top, \tag{56}$$

for an orthonormal matrix Q. The hidden activity is therefore

$$H^* = Q\,\mathrm{diag}(s_i\sqrt{a_i^*})V^\top, \tag{57}$$

and its sample kernel is

$$K_H^* := (H^*)^\top H^* = V\,\mathrm{diag}([s_i t_i - \lambda]_+)V^\top \tag{58}$$

Applying Eq. 58 to the ANOVA decomposition gives, for every nonconstant component S,

$$\boxed{k_S^{(h)} = \left[\sqrt{k_S^{\mathrm{in}} k_S^{\mathrm{out}}} - \lambda\right]_+.} \tag{59}$$

In the limit $\lambda \downarrow 0$, this proves Eq. 10. A term survives only if it is present on both sides of the task and its geometric-mean strength exceeds the regularization threshold.

Unregularized MSE trained by small-initialization gradient descent. Now set $\lambda = 0$ and the loss is

$$\mathcal{L}_0(W_1, W_2) = \|Y - W_2 W_1 X\|_F^2. \tag{60}$$

We will show that under gradient flow, which is the vanishing-step-size limit of full-batch gradient descent, the optimal hidden kernel will be Eq.59 under small regularization $\lambda \downarrow 0$. Let

$$W_1(0) = \epsilon\widetilde{W}_1, \qquad W_2(0) = \epsilon\widetilde{W}_2, \qquad \epsilon \downarrow 0, \tag{61}$$

and suppose the dynamics converge to a global least-squares solution. Writing $E = W_2 W_1 X - Y$, gradient flow obeys

$$\dot{W}_1 = -2W_2^\top E X^\top, \qquad \dot{W}_2 = -2EX^\top W_1^\top. \tag{62}$$

A direct differentiation gives the conserved quantity

$$\frac{d}{dt}\left(W_2^\top W_2 - W_1 W_1^\top\right) = 0. \tag{63}$$

Under Eq. 61, the conserved matrix is $O(\epsilon^2)$. Hence, in the infinitesimal-initialization limit, the final factors are balanced:

$$(W_2^*)^\top W_2^* = W_1^* (W_1^*)^\top. \tag{64}$$

Let $P_X = XX^\dagger$ be the orthogonal projector onto $\mathrm{Range}(X)$ in input-feature space. Equation 62 implies

$$\frac{d}{dt}\left[W_1(I - P_X)\right] = 0. \tag{65}$$

Thus $W_1(I - P_X) = O(\epsilon)$ throughout training, so its contribution vanishes as $\epsilon \downarrow 0$. The selected end-to-end least-squares map is therefore

$$A_0^* := W_2^* W_1^* = YX^\dagger. \tag{66}$$

This statement includes the nonzero error case: if an output mode is absent from the row space of X, it remains as irreducible residual error.

Let

$$A_0^* = U_A \Sigma_A V_A^\top \tag{67}$$

be its compact SVD. Equations 64 and 66 imply that

$$W_1^* = Q \Sigma_A^{1/2} V_A^\top, \qquad W_2^* = U_A \Sigma_A^{1/2} Q^\top, \tag{68}$$

up to an arbitrary orthogonal rotation Q of the active hidden subspace. Therefore the hidden kernel is unique even though the hidden coordinates are not:

$$K_{H,\mathrm{GD}}^* = X^\top \left[(A_0^*)^\top A_0^* \right]^{1/2} X. \tag{69}$$

When K_X and K_Y commute, Eq. 69 simplifies sharply. On a common mode with $s_i > 0$,

$$A_0^* u_i^X = \frac{t_i}{s_i} u_i^Y, \tag{70}$$

so the corresponding singular value of A_0^* is t_i/s_i. Substitution into Eq. 69 gives

$$K_{H,\mathrm{GD}}^* = V \,\mathrm{diag}(s_i t_i) V^\top = (K_X K_Y)^{1/2}. \tag{71}$$

Consequently, for the compositional tasks considered here,

$$\boxed{k_{S,\mathrm{GD}}^{(h)} = \sqrt{k_S^{\mathrm{in}} k_S^{\mathrm{out}}}.} \tag{72}$$

Thus small-initialization gradient descent on unregularized MSE selects exactly the $\lambda \downarrow 0$ limit of the explicitly regularized optimum.

Consequences for abstraction. Equations 59 and 72 make the rebinding effect immediate. If a component S is absent from either the input or the output, then its hidden strength is zero. In particular, every component containing an output-irrelevant factor is removed. Likewise, if a pairwise conjunction AB occurs on only one side of the task, then $k_{AB}^{(h)} = 0$ even when both individual effects are retained.

For completeness, we also derive the relation to marginal parallelism score (PS) used in the main text. After marginalizing over all factors except a target factor A and context factor B, only the components A, B, and AB can remain, apart from the mean component $S = \varnothing$. Taking a coding difference across two values of A removes the mean and B components. A direct calculation gives

$$\overline{\mathrm{PS}}(A; B) = \frac{k_A - k_{AB}}{k_A + (n-1) k_{AB}}, \tag{73}$$

provided $k_A > 0$. Equivalently,

$$\frac{k_{AB}}{k_A} = \frac{1 - \overline{\mathrm{PS}}(A; B)}{1 + (n-1)\overline{\mathrm{PS}}(A; B)}. \tag{74}$$

At finite regularization, Eq. 59 implies

$$\frac{k_{AB}^{(h)}}{k_A^{(h)}} = \frac{[\sqrt{k_{AB}^{\mathrm{in}} k_{AB}^{\mathrm{out}}} - \lambda]_+}{[\sqrt{k_A^{\mathrm{in}} k_A^{\mathrm{out}}} - \lambda]_+}, \tag{75}$$

whenever the denominator is nonzero. Eq. 10 follows directly from the definition $q = k_{AB}/k_A$ and Eq. 75 ($\lambda \downarrow 0$). Note that Eq. 74 maps these ratios to parallelism scores.

Therefore, complete rebinding has a particularly simple signature. If AB is absent from either the input or the output while A itself is shared, then $k_{AB}^{(h)} = 0$ and $\overline{\mathrm{PS}}_h(A; B) = 1$.

Pointwise context invariance under complete rebinding. The preceding results imply a stronger notion of abstraction than marginal PS. Let $\mathbf{z}_{-A}$ denote an arbitrary assignment of all factors other than A, and let $e_{a,\mathbf{z}_{-A}}$ be the trial-space basis vector for the corresponding factor combination. For two values $a \neq a'$ of A, define the sample-space contrast

$$d_{a,a'}(\mathbf{z}_{-A}) = e_{a,\mathbf{z}_{-A}} - e_{a',\mathbf{z}_{-A}}, \tag{76}$$

and the corresponding hidden coding direction

$$\Delta_{a,a'}^{(\mathbf{z}_{-A})} = H\, d_{a,a'}(\mathbf{z}_{-A}). \tag{77}$$

Suppose that A itself survives in the hidden representation, $k_A^{(h)} > 0$, but no higher-order components containing A is shared between input and output. Equivalently,

$$k_S^{\text{in}} k_S^{\text{out}} = 0 \qquad \text{for every } S \supsetneq \{A\}. \tag{78}$$

By Eq. 59, this implies $k_S^{(h)} = 0$ for every $S \supsetneq \{A\}$.

To see the consequence, apply the ANOVA projectors in Eq. 43 to an A-contrast. If $A \notin S$, then

$$\Pi_S d_{a,a'}(\mathbf{z}_{-A}) = 0, \tag{79}$$

because the A factor is projected by P_A and $P_A(e_a - e_{a'}) = 0$. For the component $S = \{A\}$,

$$\Pi_{\{A\}} d_{a,a'}(\mathbf{z}_{-A}) = (e_a - e_{a'}) \otimes \left(\frac{\mathbf{1}_n}{n}\right)^{\otimes(d-1)}, \tag{80}$$

which is independent of $\mathbf{z}_{-A}$. If all higher-order components containing A have zero hidden strength, it follows that

$$K_H\, d_{a,a'}(\mathbf{z}_{-A}) = k_A^{(h)} \Pi_{\{A\}} d_{a,a'}(\mathbf{z}_{-A}), \tag{81}$$

and the right-hand side is identical for every context.

For any two contexts $\mathbf{z}_{-A}$ and $\mathbf{z}'_{-A}$, let $u = d_{a,a'}(\mathbf{z}_{-A}) - d_{a,a'}(\mathbf{z}'_{-A})$. Equation 81 gives $K_H u = 0$. Because $K_H = H^\top H$,

$$\left\| \Delta_{a,a'}^{(\mathbf{z}_{-A})} - \Delta_{a,a'}^{(\mathbf{z}'_{-A})} \right\|^2 = u^\top K_H u = 0. \tag{82}$$

Therefore

$$\boxed{\Delta_{a,a'}^{(\mathbf{z}_{-A})} = \Delta_{a,a'}^{(\mathbf{z}'_{-A})} \quad \text{for all } \mathbf{z}_{-A}, \mathbf{z}'_{-A}.} \tag{83}$$

Thus the coding direction of A is exactly invariant across all combinations of the other factor values, not merely after averaging or marginalizing over contexts. In particular, whenever the contrast is nonzero, its PS is one for every pair of contexts.