LEARNING BY GROWING: DENDRITIC DEVELOPMENT SHAPES TASK REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

In biological brains, form and computation are intrinsically linked: dendritic growth changes both which inputs a neuron receives and how it integrates them. Standard artificial neural networks represent connections as numerical weights without modeling their physical structure. This raises a question for brain-inspired AI: can task experience guide physical growth to build useful representations? Here we train simulated biophysical neuron populations through task-guided, geometrically constrained dendritic growth and pruning. Synaptic strengths follow geometry rather than independent optimization, and a separately fitted linear readout predicts from population responses. Across synchrony and displacement tasks, growth builds predictive representations. Crucially, reassigning the same selected inputs among neurons reduces performance even when learned trees and contact strengths are preserved and the readout is retrained. This allocation advantage develops during learning. Learned populations also benefit more from combining neuronal responses than remapped populations do, supporting the value of complementary population features. More connections do not always help: in synchrony detection, active dendritic performance declines modestly at higher connectivity while a point-neuron baseline continues to improve. These findings connect biological development to representation learning, showing how task-guided growth establishes useful input organization beyond source selection. Ultimately, they reveal how intelligent representations are fundamentally rooted in their underlying physical morphology, providing a framework for investigating the formation of structural inductive biases.

1 INTRODUCTION

The architecture of an artificial neural network (ANN) shapes its inductive biases—what it can learn easily and how inputs interact. Local connectivity, weight sharing, and recurrence provide familiar examples. Neural architecture search and sparse training make aspects of network wiring learnable (Liu et al., 2019; Mocanu et al., 2018; Bellec et al., 2018). Biological development offers a complementary perspective: neural circuits form through physical growth, with activity and experience influencing which connections are established and retained (Niell et al., 2004; Xu et al., 2009). This raises a question for brain-inspired AI: can task experience guide the growth of useful computational structure? Here, we investigate this question by training simulated populations of biophysical neurons through dendritic growth and pruning, without independently optimizing synaptic weights.

Dendrites provide a concrete link between biological development and computational architecture. A standard point neuron applies an activation function to a weighted sum of inputs. Biological neurons receive and process signals through branching dendrites that act as electrical cables. Extending a branch can change both which inputs a neuron reaches and how their signals interact before reaching the cell body (Mainen & Sejnowski, 1996; London & Häusser, 2005). Input connectivity and signal integration therefore depend on the same physical structure (Figure 1a–b). Growing dendrites changes both the evidence available to a neuron and how that evidence is processed. This raises a further question: does access to more inputs necessarily produce more useful representations?

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.

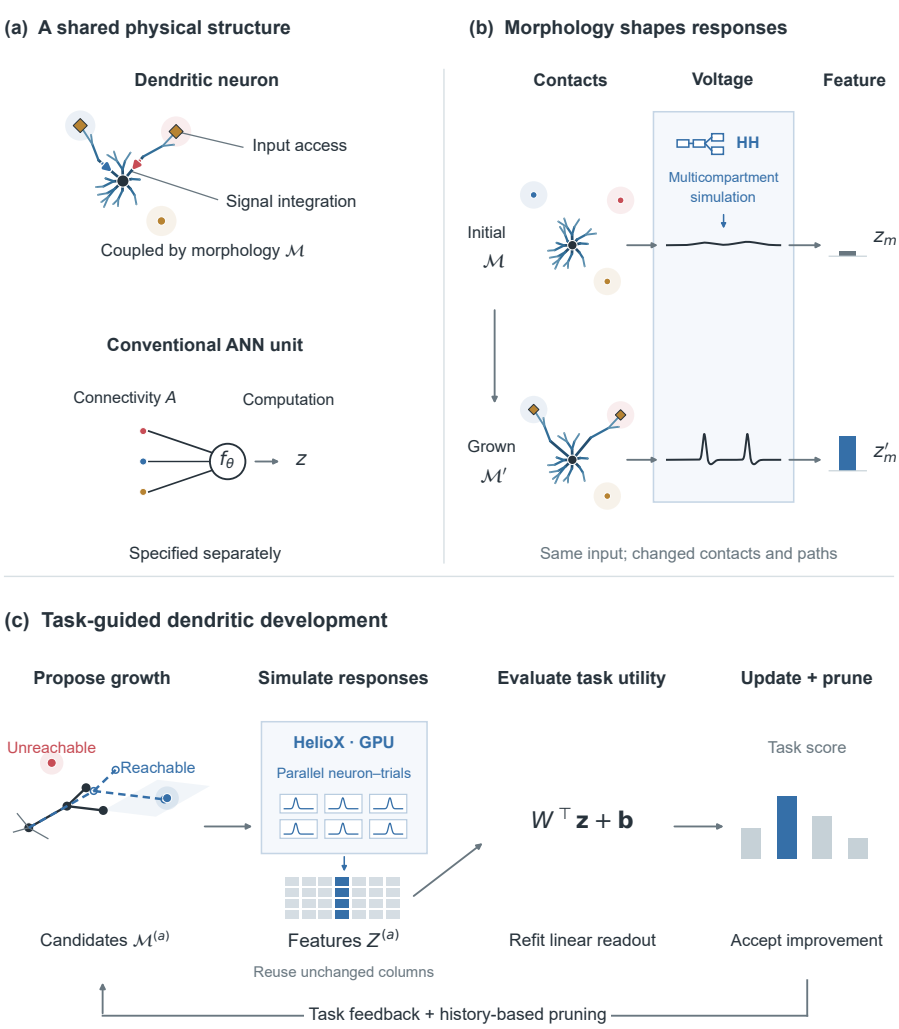

Figure 1: **Learning by developing the structures that receive and process inputs.** (a) Dendritic morphology $\mathcal{M}$ couples input access and signal integration. (b) Growth changes contacts and their strengths, multicompartment HH responses, and neuronal feature z. (c) The learning loop proposes feasible growth, simulates candidate responses, refits a readout to evaluate task utility, and accepts improving changes with history-based pruning. HelioX (Lu et al., 2026) parallelizes simulation; unchanged feature columns are reused. Traces are schematic.

A central challenge is understanding what useful organization growth establishes. Previous models have optimized dendritic morphology for specific computations and studied growth through activity-dependent plasticity (Torben-Nielsen & Stiefel, 2010; Kirchner et al., 2025). Improved prediction alone, however, does not reveal what structural learning has achieved. A growing population might acquire informative inputs (source selection), or it might also learn useful assignments of those inputs to receiving neurons (input allocation). The same selected sources can produce different population features depending on where their signals are received and combined. We distinguish these contributions by redistributing selected sources among developed neurons and testing whether a retrained readout can recover the original performance. Repeating this intervention during growth reveals when learned assignments become useful.

Another challenge is the high computational cost of simulating biophysically detailed neurons with dendrites and ion channels. Structural learning requires simulating the responses of many candidate morphologies to task inputs, repeatedly throughout development. Even small populations require millions of neuronal response evaluations over a training trajectory. This simulation-intensive search

is made practical by utilizing HelioX (Lu et al., 2026), a GPU-native framework for highly parallel biophysical simulation, and reusing valid responses from unchanged neurons. Task feedback guides geometrically constrained growth and pruning through predictions from a separately fitted linear readout (Figure 1c). Synaptic strengths follow the changing geometry, with fixed conductance scales and membrane parameter densities.

We evaluate the framework on majority synchrony detection, displacement direction detection, and continuous displacement estimation (hereafter synchrony, direction, and displacement). Synchrony detection requires combining temporal evidence despite distractors. The two spatial tasks require relating changes in activity to spatial position to predict movement direction and extent. These controlled tasks provide known sources of temporal and spatial evidence, allowing us to connect where dendrites grow, which inputs they receive, and how the resulting responses support prediction.

Our main contributions are:

1. **Learning representations by growing dendrites.** Task-guided growth and pruning build predictive population representations without independent synaptic-weight optimization. Across all three tasks and separately trained spatial arrangements, prediction improves while neuron number, membrane parameter densities, and linear readout capacity remain fixed (Figure 2).

2. **Input allocation matters beyond source selection.** Reassigning the same selected sources among developed neurons reduces performance even when dendritic trees and contact strengths are preserved and the readout is retrained. This allocation advantage emerges during growth. Compared with remapped populations, learned populations also show larger predictive gains from combining neurons relative to using the best single neuron, supporting the value of complementary population features (Figure 3).

3. **Challenging the "more is better" assumption for input connectivity.** Additional connections do not always improve prediction. In synchrony detection, active dendritic performance declines modestly at higher connection budgets while the sparse point-neuron comparator continues to improve. Together with the remapping results, this shows that input availability alone is insufficient to explain the usefulness of the learned representation: how inputs are assigned and transformed also matters (Figure 4).

These findings connect dendritic development to representation learning and raise a concrete question for adaptive and sparse architectures: which inputs should be processed together, and what features do those combinations produce? Our framework makes this question testable under physical growth constraints, providing a setting for investigating how task experience contributes to the formation of structural inductive biases.

2 Learning through dendritic growth

2.1 Framework overview

We train $N = 16$ biophysical neurons by developing dendritic trees $\mathcal{M}$ around spatially fixed input sources. The framework alternates growth proposals, response simulation, readout-based evaluation, and structural updates with pruning (Figure 1c; Algorithm 1). Task feedback thus shapes the structures that receive and process information. Neuron number, membrane parameter densities, and readout capacity remain fixed.

Algorithm 1 Task-guided dendritic development

Initialize: trees $\mathcal{M}$, fixed source locations, training data, and growth history.

Repeat until the structural budget or stopping criterion is reached:

1. Propose. Generate feasible branch extensions using task information, current responses, and exploration; include a no-change option.

2. Simulate. Reconstruct contacts and weights; simulate changed neurons in parallel and reuse unchanged responses to assemble candidate features.

3. Evaluate. Refit a linear readout for each candidate and score its prediction using training-side data splits.

4. Update and prune. Accept the best strictly improving candidate or retain the current trees. Record growth; periodically reevaluate and revise earlier additions.

Return: developed trees and the population representation they produce.

Updates extend and activate branches within prescribed planar templates; extending an ancestor can also move downstream branches and change multiple contacts. Figures 5–6 follow a complete primary-budget trajectory, showing how trees, contact sites, and coupling strengths develop together.

2.2 Geometry couples input access and signal integration

Synaptic strengths follow geometry. Bringing a dendrite closer to a source increases its coupling. For a retained, unmerged contact c from source i, the implemented rule is

$$w_{ic}(\mathcal{M}) = \bar{w}_i \exp\left[-\frac{p\, d_{ic}(\mathcal{M})^2}{2\sigma_i^2}\right], \tag{1}$$

where d_{ic} is source–contact distance, σ_i the source's spatial width, p a fixed exponent, and $\bar{w}_i$ a fixed conductance scale. Growth therefore changes effective synaptic weights without optimizing them independently. Contact eligibility and aggregation are specified in Appendix A.

The same geometry performs the computation. Each tree is discretized into electrically coupled compartments: cable segments with their own membrane voltages. For compartment j of neuron m, the biophysical forward pass integrates

$$C_{mj}(\mathcal{M})\dot{V}_{mj} = -I^{\mathrm{mem}}_{mj}(V_{mj}, \mathbf{h}_{mj}; \mathcal{M}) + \sum_{k\sim j} G_{mjk}(\mathcal{M})(V_{mk} - V_{mj}) + I^{\mathrm{syn}}_{mj}(t; x, \mathcal{M}). \tag{2}$$

Morphology determines compartment capacitances C, axial conductances G, and the placement and strength of synaptic inputs I^{syn}. The HH model includes Hodgkin–Huxley sodium, potassium, and leak currents with channel states $\mathbf{h}$ (Hodgkin & Huxley, 1952); Passive retains cable coupling and synaptic input with leak-only membrane currents. Thus growth changes both which signals enter and how they interact, even with fixed membrane parameter densities.

2.3 Task feedback guides structural search

Evaluate the representation, then update its structure. Each neuron contributes a scalar feature z_m: a change in somatic spike count (HH) or voltage (Passive) for direction and displacement, or a voltage-presence statistic for synchrony. The neurons receive external inputs independently; a linear readout, $\widehat{\mathbf{y}} = W^\top \mathbf{z} + \mathbf{b}$, combines their responses. For candidate actions $\mathcal{A}_t$, including no change, selection is

$$a_t^* \in \arg\min_{a\in\mathcal{A}_t} L_{\mathrm{train}}(Z(\mathcal{M}_t \oplus a), \mathbf{y}), \tag{3}$$

where Z contains features across examples, $\oplus a$ applies a geometric action, and L_{train} is the cross-fitted prediction loss of a refitted readout. Structural decisions exclude the report-only observer and require no simulator gradients (Appendices A.5–B.3).

Parallel simulation makes repeated evaluation practical. Candidate evaluation repeatedly integrates multicompartment dynamics. A single-neuron proposal changes one column of Z, allowing unchanged responses to be reused on the same inputs. With n_0 current-state trials and U candidate neuron configurations evaluated on n_c trials each, the nominal work is

$$\mathcal{W}_{\mathrm{round}} = Nn_0 + Un_c, \qquad \mathcal{W}_{\mathrm{search}} = \sum_t \mathcal{W}_{\mathrm{round},t}. \tag{4}$$

Reuse reduces the candidate term from NUn_c to Un_c. Complete training records account for 20.30, 4.49, and 9.13 million neuron–trials for synchrony, direction, and displacement, before reuse of identical or cached responses. HelioX (Lu et al., 2026) batches independent configurations and trials on GPUs while structural updates remain sequential. The candidate-evaluation pipeline uses device-side event delivery, task-specific batching, and memory reuse (Appendix A.6).

3 Experimental design

Tasks and evaluation. We ask whether dendritic growth can learn temporal and spatial relationships (Figure 2a–c). *Majority synchrony* detects coordinated event timing amid distractors. *Direction detection* classifies the direction of a shift in source activity. *Displacement estimation* predicts its continuous two-dimensional displacement. Sources stay physically fixed. We report balanced accuracy, accuracy, and mean coordinate R^2, respectively. Each task has four worlds under the same initialization rules, including the illustrated world, and four training seeds per physical implementation. A separate observer fits and evaluates the readout on data excluded from structural decisions. Curves average worlds equally; bands show their range. The data partitions and observer protocol are specified in Appendix B.

Mechanistic tests. We freeze learned trees and contact strengths, remap source identities, and refit the readout. *Source-held* remapping preserves the selected sources while changing their allocation among neurons; *full-graph* remapping also changes source selection. Each uses eight remaps of one HH trajectory per world. This tests what is lost when the learned input organization is disrupted.

Connectivity comparison. HH and Passive are compared with a sparse *Point* model containing 16 tanh units. All produce 16 features for the same task-specific readout rule. Physical access is measured by an equivalent connection budget B_{eq}, which counts strong source–neuron couplings once and weak ones fractionally; Point uses the number K of nonzero source–unit groups. We examine the primary budget range through 48. The models share evaluation examples but differ in input encoding and training. The comparison tests how predictive performance changes with input connectivity within each implementation. Exact budgets, splits, and interventions are given in Appendices A and B.

4 Results

4.1 Learning representations by growing dendrites

Task experience leaves a visible signature in the dendritic trees. Starting from the same illustrated population, different tasks produce different patterns of growth and source contact (Figure 2d–f). The highlighted neuron makes this adaptation easy to follow: its branches develop around the task's inputs, and its voltage response changes even when the stimulus is held constant (panels g–i). Morphological development changes what the neuron receives and the response it contributes to the population.

These changes build useful representations across worlds (Figure 2j–l). Mean synchrony balanced accuracy rises from 61.9% to 80.4%, direction accuracy from 46.1% to 98.7%, and displacement R^2 from 0.571 to 0.860, from initialization to the nominal-48 endpoint. Neuron number and readout capacity remain unchanged. The representational change comes from developing the physical input-processing structure, including its geometry-derived weights. We next separate two contributions to this improvement: selecting useful inputs and learning how to combine them.

4.2 Input allocation matters beyond source selection

Selecting the right sources is not enough to preserve the learned representation. Source-held remapping retains those sources and the developed trees, but changes which neurons receive which sources (Figure 3a). The intervention also preserves contact sites and strengths and the strong-contact degrees of both sources and neurons. It changes which event streams reach which receiving cells. Performance falls even after readout retraining, locating the loss in the remapped representation.

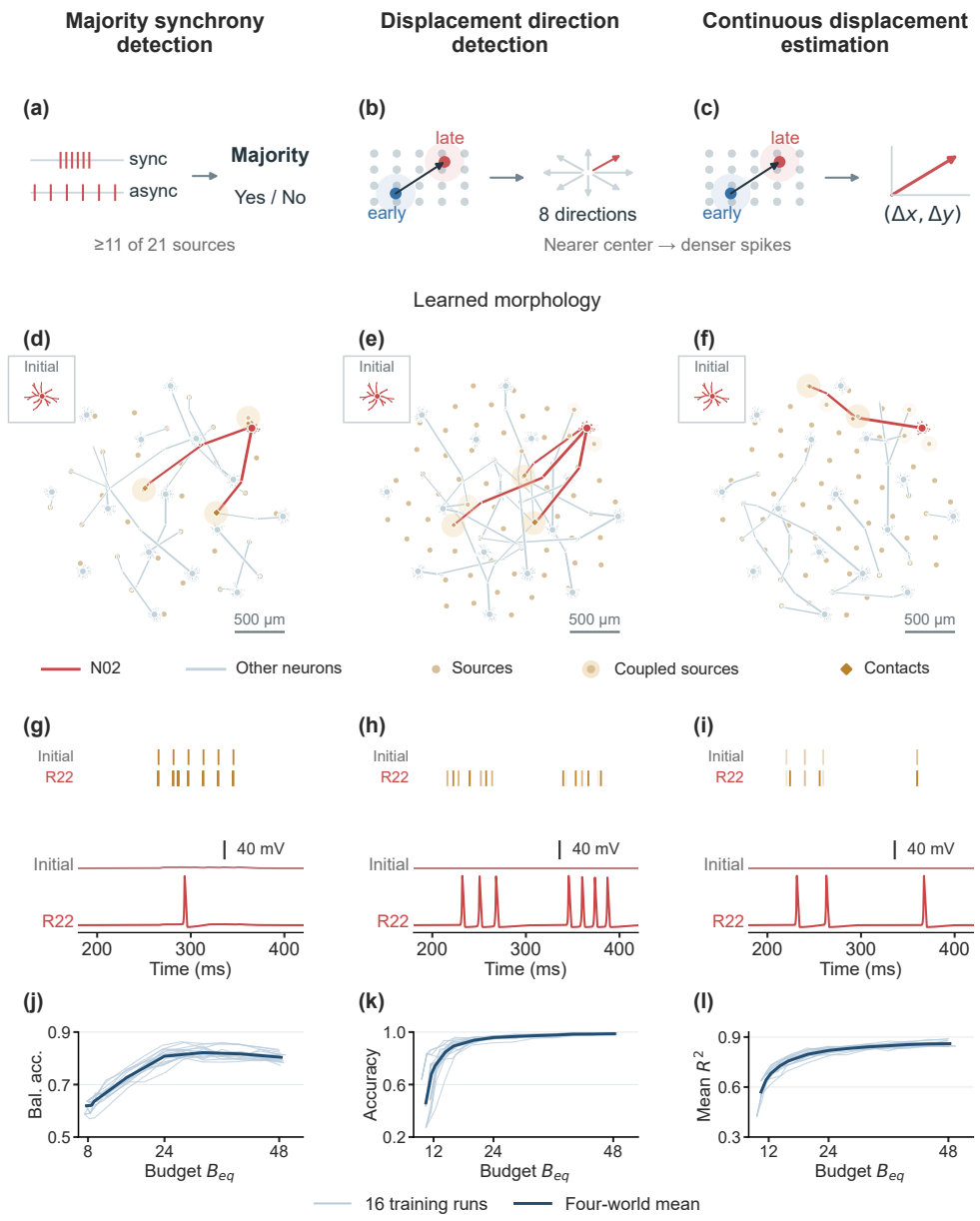

Figure 2: **Task-guided growth changes contacts and neuronal responses.** (a–c) Task designs. (d–f) Original-world arbors at round 22, with N02 highlighted and its initial arbor inset; diamonds mark contacts and halos indicate source coupling. (g–i) Coupled events and somatic HH voltage before/after growth under the same fitting stimulus; traces are offset (40 mV bars). (j–l) Observer performance for four training seeds in each of four worlds, including the illustrated world. Thin lines show runs; dark lines average world means, using measured budgets and true initial evaluations.

This effect appears in every world for all three tasks at the endpoint (Figure 3b–d). Mean losses are 4.1 and 4.9 percentage points for synchrony and direction, and 0.17 R^2 for displacement. Full-graph remapping causes larger mean losses: 11.1 and 11.7 points, and 0.19 R^2, respectively. Growth therefore learns a useful allocation of inputs in addition to finding informative sources: which inputs meet within a neuron matters to the population's prediction. The transition from near-zero or negative early effects to positive endpoint effects traces the acquisition of useful input allocation during development (Appendix E).

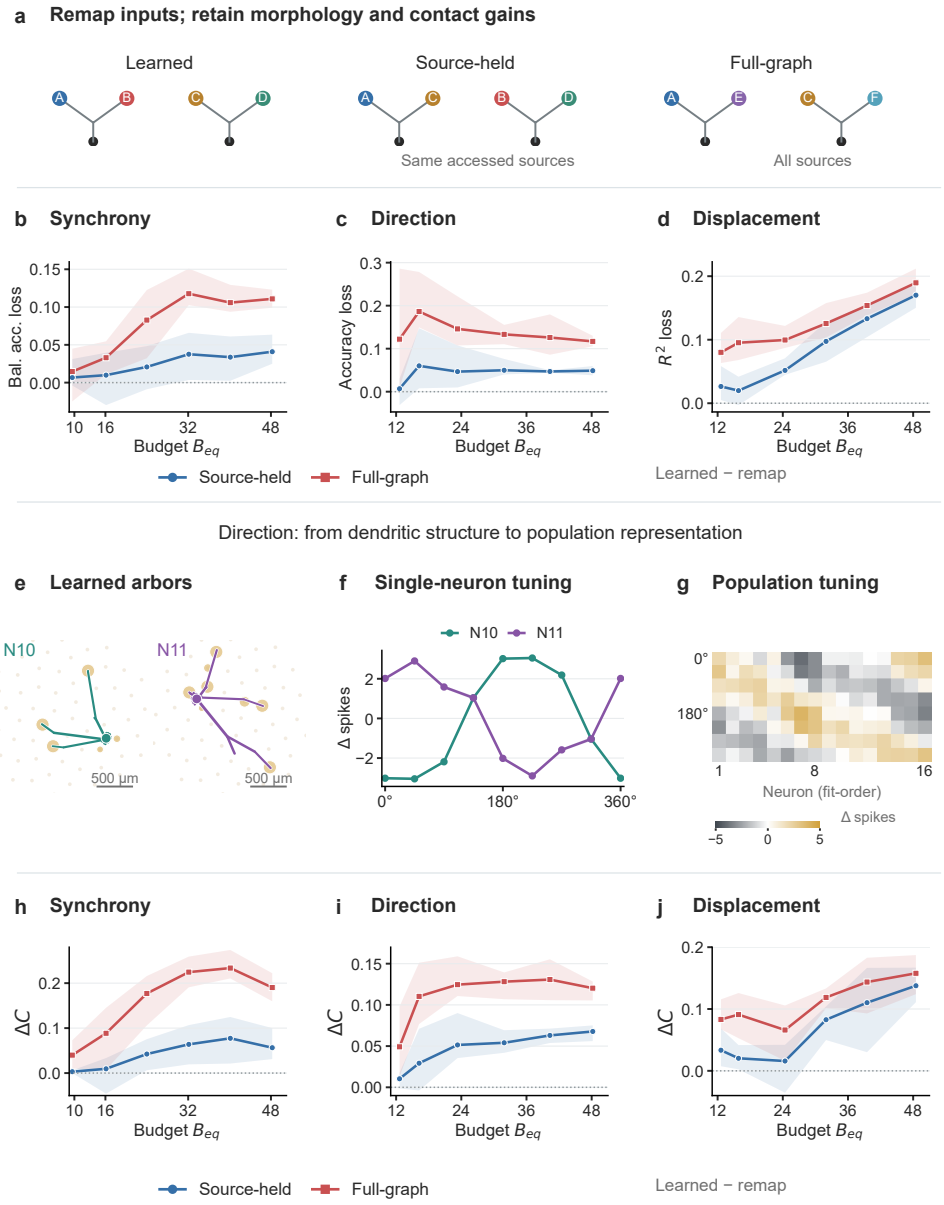

Figure 3: **Learned input allocation supports complementary population features.** (a) Remapping interventions. (b–d) Learned performance minus the median of eight remaps per family. (e–g) Direction arbors N10/N11, tuning (mean ± trial SEM), and learned population responses at $B_{\mathrm{eq}} = 48.06$; fitting data determine pair selection and heatmap order. (h–j) Learned-minus-remapped population gain, ΔC, comparing the benefit of combining neurons over the best single neuron (Appendix E). Lines/bands show the mean/full range of four paired world effects, one HH trajectory per world. Budget anchors start near $B_0 + 2$; actual budgets are plotted. Complete early curves appear in Appendix E.

4.3 Learned allocation supports complementary population features

Direction reveals what this organization looks like. Two developed neurons receive different spatial combinations of inputs and prefer markedly different directions (Figure 3e–f), with preferred phases separated by approximately 158°. The population heatmap extends this picture: neurons respond

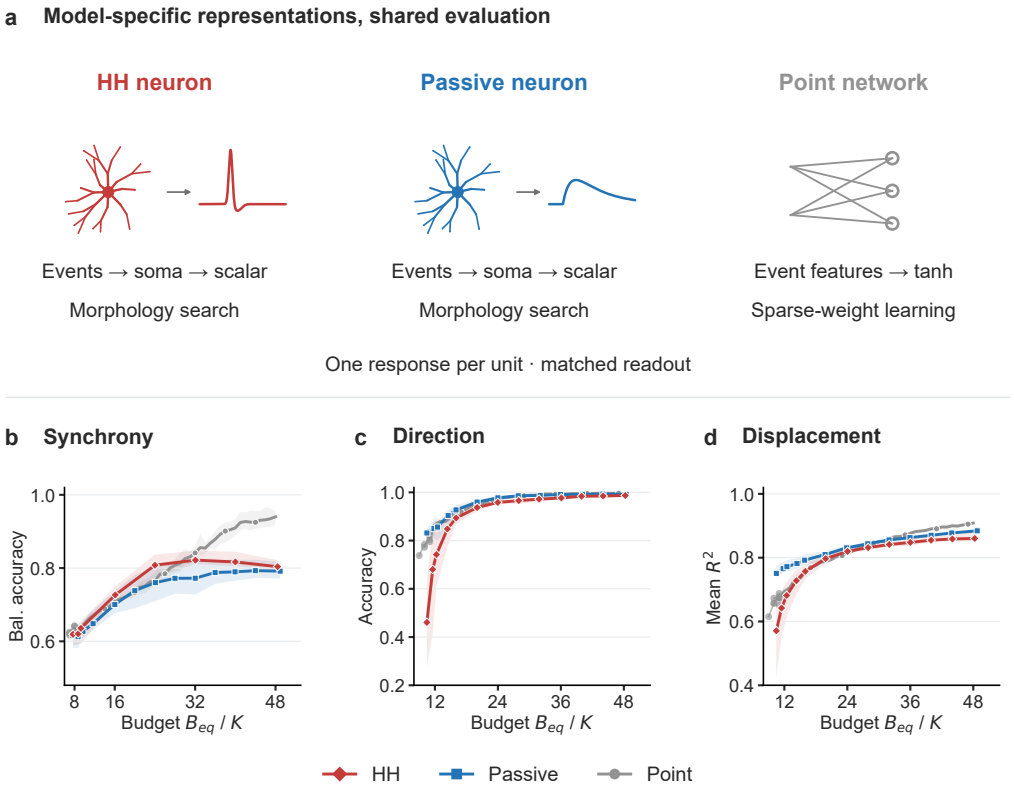

Figure 4: **More input connections do not guarantee better population representations.** (a) Implementations. (b–d) Observer performance versus physical B_{eq} or Point K, including true initial physical states. Lines/bands: mean/full range of four world means (four physical or eight Point seeds per world). Short early Point traces show individual worlds. Observer samples and readout capacity are shared; input encodings and training differ.

differently to the same directions, distributing task information across a set of tuned responses (panel g). Growth develops a population whose members contribute different views of the same input.

To test whether these responses help each other, we measure *population gain*: the reduction in held-out prediction error obtained by combining neurons instead of using the best individual neuron, normalized by the error of a constant predictor. This links the tuning differences to their predictive contribution: combining responses can recover task information beyond the strongest individual feature. The same measure applies to all three tasks. Learned populations have greater gain than source-held remaps in every world at the endpoint (Figure 3h–j). The mean learned-minus-remapped advantage, ΔC, is 0.056 for synchrony, 0.068 for direction, and 0.138 for displacement. Normalization expresses the added predictive value on a common scale across tasks. Thus the allocation learned during growth improves what the neurons can predict together, not just which sources they access. Appendix E gives the estimator and full budget trajectories.

4.4 More input connections do not guarantee better representations

Does a growing network simply need more connections? The Point baseline follows the intuitive trend: denser input connectivity improves performance overall on each task (Figure 4). The dendritic populations challenge that expectation. In synchrony, HH performance peaks near budget 32 and then declines, even as input access increases.

Over the same interval, Point balanced accuracy rises from 84.1% to 94.0%, while HH falls from 82.2% to 80.4%. More access can therefore accompany a less predictive physical representation.

Direction approaches the task ceiling, with endpoint accuracies of 98.7–99.5% across implementations. Displacement continues to benefit from additional connections: endpoint R^2 is 0.860 for HH, 0.884 for Passive, and 0.909 for Point. Additional connectivity remains useful for displacement, but it is insufficient to guarantee better representations across tasks. Its benefit depends on the responses produced by the developed structure.

In a point unit, an additional input is combined through an independently adjustable weight. In a dendritic neuron, growth changes both input weights and the cable structure that integrates those inputs. Acquiring an input and changing the receiving computation are coupled operations. Together, the remapping and budget results show why input availability alone is insufficient: how inputs are assigned and transformed also matters.

5 Related work

Learning network structure. Neural architecture search optimizes computational graphs (Liu et al., 2019); sparse training adapts connectivity while learning weights (Mocanu et al., 2018; Bellec et al., 2018). Developmental programs can also construct neural networks through growth-like rules (Najarro et al., 2023). In our setting, physical dendritic geometry determines source coupling and each neuron's internal computation, making the learned architecture a spatial input-processing structure.

Dendritic computation and morphology. Dendrites support computations beyond a single weighted sum (London & Häusser, 2005; Poirazi et al., 2003; Beniaguev et al., 2021), and morphology can alter neuronal response dynamics (Mainen & Sejnowski, 1996). Dendritic shape has been optimized for specific computations (Torben-Nielsen & Stiefel, 2010), and developmental models couple growth to activity-dependent plasticity (Kirchner et al., 2025). Our focus is how task-guided development organizes a population representation. Counterfactual source remapping separates the value of selected inputs from their allocation across developed neurons, and population gain measures the predictive benefit of combining the resulting responses.

Learning with physical dynamics. Physical neural networks train controllable substrates to perform useful transformations (Wright et al., 2022; Dillavou et al., 2022). Biophysical simulators support learning: Jaxley supports differentiable training (Deistler et al., 2025), while HelioX provides GPU-native simulation and training (Lu et al., 2026). The physical system here is simulated, and its geometry-derived synaptic weights change through growth rather than independent parameter optimization.

6 Discussion

Dendritic development gives architecture learning a physical form: the structure that reaches an input also helps compute with it. The remapping experiments show that useful development includes learning where inputs belong. The population-gain analysis links this assignment to complementary task representations across temporal and spatial problems.

This result connects adaptive connectivity and developmental network construction (Mocanu et al., 2018; Bellec et al., 2018; Najarro et al., 2023) with dendritic computation and optimized morphology (Poirazi et al., 2003; Torben-Nielsen & Stiefel, 2010). Like work on physical learning (Wright et al., 2022; Dillavou et al., 2022), it treats the substrate as part of the computation. Whereas differentiable biophysical simulation supports parameter learning (Deistler et al., 2025), our search adapts geometry, with synaptic weights determined by the evolving structure and membrane parameter densities held fixed.

The present system isolates structural learning through centralized supervision, prescribed branch directions, and controlled tasks across four worlds. Remapping probes input allocation in fixed developed trees; model-specific connectivity curves reveal how each implementation uses additional access. These findings motivate richer task families, more flexible growth, and local structural-learning rules. The central architecture question is which inputs should be processed together and what features those combinations produce. Task-guided development provides a setting for investigating how these choices shape structural inductive biases.

AI USE STATEMENT

Generative AI tools assisted with conceptual framing, experimental planning, code development, analysis and interpretation, figure preparation, literature organization, and manuscript drafting and revision. For synthetic task data, AI assistance concerned the design and code for programmatically generating task inputs. Neuronal responses were generated by numerical simulation of the specified biophysical models, and performance and representation statistics were computed from those simulation outputs. The authors take responsibility for the study design, verification of data and references, and final manuscript.

REPRODUCIBILITY STATEMENT

Section 2 specifies the physical computation and learning loop. Appendices A and B give geometry, contact, response, readout, training, and intervention definitions. Appendix D reports the four-world aggregation and endpoint measurements; Appendix E gives the population-gain estimator, figure-selection rules, and supplementary response plots. The analysis uses frozen geometry and evaluation records with recorded content hashes. Training and analysis code, frozen figure data, and a CPU reproduction example are provided in the supplementary material, together with a precompiled HelioX runtime and platform-specific setup instructions.

REFERENCES

Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep Rewiring: Training Very Sparse Deep Networks. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1711.05136.

D. Beniaguev, I. Segev, and M. London. Single cortical neurons as deep artificial neural networks. *Neuron*, 109:2727–2739.e3, 2021. doi: 10.1016/j.neuron.2021.07.002. URL https://doi.org/10.1016/j.neuron.2021.07.002.

R. D. Cazé, M. Humphries, and B. Gutkin. Passive dendrites enable single neurons to compute linearly non-separable functions. *PLOS Computational Biology*, 9:e1002867, 2013. doi: 10.1371/journal.pcbi.1002867. URL https://doi.org/10.1371/journal.pcbi.1002867.

Michael Deistler, Kyra L. Kadhim, Matthijs Pals, Jonas Beck, Ziwei Huang, Manuel Gloeckler, Janne K. Lappalainen, Cornelius Schröder, Philipp Berens, Pedro J. Gonçalves, and Jakob H. Macke. Jaxley: differentiable simulation enables large-scale training of detailed biophysical models of neural dynamics. *Nature Methods*, 22:2649–2657, 2025. doi: 10.1038/s41592-025-02895-w. URL https://doi.org/10.1038/s41592-025-02895-w.

S. Dillavou, M. Stern, A. J. Liu, and D. J. Durian. Demonstration of decentralized physics-driven learning. *Physical Review Applied*, 18:014040, 2022. doi: 10.1103/PhysRevApplied.18.014040. URL https://doi.org/10.1103/PhysRevApplied.18.014040.

A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117:500–544, 1952. doi: 10.1113/jphysiol.1952.sp004764. URL https://doi.org/10.1113/jphysiol.1952.sp004764.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980.

Jan H. Kirchner, Lucas Euler, Ingo Fritz, André Ferreira Castro, and Julijana Gjorgjieva. Dendritic growth and synaptic organization from activity-independent cues and local activity-dependent plasticity. *eLife*, 12:RP87527, 2025. doi: 10.7554/eLife.87527.3. URL https://doi.org/10.7554/eLife.87527.3.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://arxiv.org/abs/1806.09055.

M. London and M. Häusser. Dendritic computation. *Annual Review of Neuroscience*, 28:503–532, 2005. doi: 10.1146/annurev.neuro.28.061604.135703. URL https://doi.org/10.1146/annurev.neuro.28.061604.135703.

J. Lu, Z. Yu, S. Cui, G. He, R. Xiong, K. Du, and T. Huang. HelioX: A GPU-native framework for simulation and training of biophysically detailed networks. In *International Conference on Machine Learning (ICML)*, 2026. URL https://openreview.net/forum?id=XjTRtfxK9g.

Z. F. Mainen and T. J. Sejnowski. Influence of dendritic structure on firing pattern in model neocortical neurons. *Nature*, 382:363–366, 1996. doi: 10.1038/382363a0. URL https://doi.org/10.1038/382363a0.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 2018. doi: 10.1038/s41467-018-04316-3. URL https://doi.org/10.1038/s41467-018-04316-3.

Elias Najarro, Shyam Sudhakaran, and Sebastian Risi. Towards Self-Assembling Artificial Neural Networks through Neural Developmental Programs. arXiv:2307.08197, 2023. URL https://arxiv.org/abs/2307.08197.

Cristopher M. Niell, Martin P. Meyer, and Stephen J. Smith. In vivo imaging of synapse formation on a growing dendritic arbor. *Nature Neuroscience*, 7:254–260, 2004. doi: 10.1038/nn1191. URL https://doi.org/10.1038/nn1191.

P. Poirazi, T. Brannon, and B. W. Mel. Pyramidal neuron as two-layer neural network. *Neuron*, 37:989–999, 2003. doi: 10.1016/S0896-6273(03)00149-1. URL https://doi.org/10.1016/S0896-6273(03)00149-1.

B. Torben-Nielsen and K. M. Stiefel. An inverse approach for elucidating dendritic function. *Frontiers in Computational Neuroscience*, 4:128, 2010. doi: 10.3389/fncom.2010.00128. URL https://doi.org/10.3389/fncom.2010.00128.

L. G. Wright, T. Onodera, M. M. Stein, T. Wang, D. T. Schachter, Z. Hu, and P. L. McMahon. Deep physical neural networks trained with backpropagation. *Nature*, 601:549–555, 2022. doi: 10.1038/s41586-021-04223-6. URL https://doi.org/10.1038/s41586-021-04223-6.

T. Xu, X. Yu, A. J. Perlik, W. F. Tobin, J. A. Zweig, K. Tennant, T. Jones, and Y. Zuo. Rapid formation and selective stabilization of synapses for enduring motor memories. *Nature*, 462:915–919, 2009. doi: 10.1038/nature08389. URL https://doi.org/10.1038/nature08389.

A PHYSICAL MODEL AND STRUCTURAL LEARNING

A.1 LEARNED VARIABLES AND GEOMETRY

Example x specifies source events $E_i(x) = \{t_{ik}\}_k$ and target $y(x)$. Source positions $\mathbf{p}_i$ and soma positions $\mathbf{q}_m$ are fixed within a world. The morphology $\mathcal{M}$ contains the active tree and lengths ℓ_{mj} of its segments. Segment directions $\mathbf{u}_{mj}$ are prescribed by the initialized tree. For a segment with parent $p(j)$,

$$\mathbf{a}_{mj} = \begin{cases} \mathbf{q}_m, & j \text{ is soma-rooted,} \\ \mathbf{b}_{m,p(j)}, & \text{otherwise,} \end{cases} \qquad \mathbf{b}_{mj} = \mathbf{a}_{mj} + \ell_{mj}\mathbf{u}_{mj}. \tag{5}$$

Extending an ancestor moves its descendants even if their lengths are unchanged. Growth activates and extends branches subject to the prescribed minimum-length and tree constraints; it does not freely rotate branches or install arbitrary direct source-to-soma edges. Pruning revises recorded structural additions. The learning variables are therefore a restricted family of geometries, not all possible graphs.

Source positions, soma positions, branch templates, and their initialization distributions are paired across physical implementations within a world. New worlds vary sampled positions and initial trees under the same task-specific spatial coverage and initialization rules as the original. The original world is included in every four-world statistical summary. The population always contains 16 receiving neurons. Their independent somatic features are combined by the readout; the experiment does not learn recurrent connections between receiving neurons.

A.2 DISTRIBUTED CONTACTS AND EQUIVALENT ACCESS

Eligible terminal segments are sampled for contacts at positions $\mathbf{r}_c$. Raw access from source i and retained-contact gain take the form

$$a_{ic} = \exp\left(-\frac{\|\mathbf{r}_c - \mathbf{p}_i\|^2}{2\sigma_i^2}\right), \qquad g_{ic} = \left[\max(a_{ic}, a_{\min})\right]^p \omega_c. \tag{6}$$

Widths, inclusion thresholds, the floor $a_{\min}$, and exponent p belong to the fixed task configuration. The multiplier ω_c accounts for contact aggregation. Contacts from the same source on a segment can be merged at their effective-gain-weighted location while preserving summed gain. The simulator evaluates dynamics at the resulting merged contact locations.

The synaptic weight is $w_{ic}(\mathcal{M}) = \bar{w}_i g_{ic}(\mathcal{M})$, where $\bar{w}_i$ is a fixed source/type-specific conductance scale. The implemented gain floor is zero and each unmerged contact has $\omega_c = 1$, giving Eq. 1. Moving such a contact closer to its source increases its weight. Merged contacts preserve the sum of their constituent gains. Synaptic weights therefore change with morphology; they are not independent optimization variables. The equivalent budget is

$$B_{\mathrm{eq}}(\mathcal{M}) = \sum_{m=1}^{N} \sum_{i} \min\left(1, \sum_{c \in \mathcal{C}_{im}} g_{imc}(\mathcal{M})\right), \tag{7}$$

It sums contact gains within each source–neuron pair and caps that sum at one for accounting purposes. Raw contact count, strongly connected pair count, cable length, and B_{eq} are consequently different measurements. Pruning can reduce the current budget; learning round and historical maximum budget cannot substitute for its actual value.

A.3 MULTICOMPARTMENT BIOPHYSICS

The simulator discretizes the soma and dendritic segments into electrically coupled compartments and integrates Eq. 2. In an HH compartment of membrane area A_{mj}, the membrane current has the conductance-based form

$$I_{mj}^{\mathrm{mem}} = A_{mj}\left[\bar{g}_{\mathrm{Na}}\, u^3 h (V_{mj} - E_{\mathrm{Na}}) + \bar{g}_{\mathrm{K}}\, n^4 (V_{mj} - E_{\mathrm{K}}) + g_{\mathrm{L}}(V_{mj} - E_{\mathrm{L}})\right], \tag{8}$$

$$\dot{q} = \alpha_q(V_{mj})(1 - q) - \beta_q(V_{mj})q, \qquad q \in \{u, h, n\}. \tag{9}$$

We use u for sodium activation to avoid confusion with the neuron index m. The channel mechanism supplies the HH voltage-dependent rates (Hodgkin & Huxley, 1952); the configured channel densities and reversal parameters are fixed during learning. Capacitance is specific membrane capacitance times area. Axial conductance follows the connected cable geometry and axial resistivity. Changing length can therefore alter current propagation, membrane area, and total capacitance even under fixed parameter densities. Passive replaces the active sodium/potassium mechanism with leak while retaining geometry and synaptic driving currents.

Source events drive double-exponential conductance synapses at the contact compartments. Writing k_c for the configured causal synaptic kernel and $w_c(\mathcal{M})$ for effective conductance,

$$I_{mj}^{\mathrm{syn}}(t) = \sum_{c \to (m,j)} w_c(\mathcal{M}) \sum_{t_{ik} \in E_{i(c)}} k_c(t - t_{ik}) \, [E_c - V_{mj}(t)]. \tag{10}$$

Thus inputs influence voltage through both contact gain and a voltage-dependent driving force. Passive cable computation with such synapses is not equivalent to simply summing a linear input feature vector (Cazé et al., 2013). HelioX integrates the numerical compartment system; no fitted ANN or analytic surrogate supplies the candidate responses.

Table 1: Fixed simulation and contact settings verified against the experiment configurations and backend. Conductance densities are held fixed as membrane area changes. Synaptic coefficients are multiplied by geometry-induced gains.

Setting	Value
Integration step; temperature	0.1 ms; 6.3°C
HH initialization; leak reversal	-65 mV; -65 mV
HH sodium / potassium density	0.12 / 0.036 S cm^{-2}
Leak conductance density	3×10^{-5} S cm^{-2}
Dendritic compartments per section	5
Base dendritic diameter; level factor	1 μm; 0.9
Maximum section length	700 μm
Synaptic rise / decay; reversal	0.1 / 10 ms; 0 mV
Conductance scale, synchrony	9×10^{-5} μS
Conductance scale, spatial tasks	7×10^{-4} μS
Source width σ; gain power p	58 μm; 3
Contact samples per eligible section	5, over relative positions $[0.15, 0.95]$
Raw-access inclusion threshold; gain floor	0.005; 0

A.4 One scalar response per receiving neuron

All three implementations provide 16 scalar features to the final readout. Physical features are computed from somatic voltage. For synchrony, let $\mathcal{T} = [220, 400)$ ms. The smooth historical-presence feature is

$$z_m = 1 - \exp\left[-\frac{\Delta t}{1.2\,\mathrm{ms}} \sum_{t \in \mathcal{T}} \mathrm{sigmoid}\left(\frac{V_m(t) - c_\nu}{s_\nu} \right) \right]. \tag{11}$$

HH uses $c_{\mathrm{HH}} = 0$ mV and $s_{\mathrm{HH}} = 5$ mV. Passive uses the calibrated constants $c_{\mathrm{Passive}} = -45.019199$ mV and $s_{\mathrm{Passive}} = 3.566488$ mV. Matrix assembly subtracts each neuron's mean blank-probe response before readout fitting. These transforms define the implementation-specific features supplied to the shared readout protocol.

Both spatial tasks use early $\mathcal{T}_E = [210, 270)$ ms and late $\mathcal{T}_L = [330, 390)$ ms windows. HH provides

$$z_m^{\mathrm{HH}} = N_m(\mathcal{T}_L) - N_m(\mathcal{T}_E), \tag{12}$$

where N_m counts somatic upward crossings of 0 mV. Passive uses $\widetilde{V}_m$, the input-driven voltage minus the no-input response of the same morphology:

$$z_m^{\mathrm{Passive}} = \frac{Q_{.99}(\widetilde{V}_m|_{\mathcal{T}_L}) - Q_{.99}(\widetilde{V}_m|_{\mathcal{T}_E})}{s_{\mathrm{task},\nu}}. \tag{13}$$

The denominator is a frozen training-derived scale. There is no tanh transform, clipping, or window-mean smoothing in this physical response definition. These implementation-specific measurements are distinct from the shared downstream readout.

A.5 Proposal, confirmation, and revision

For tested feasible changes $\mathcal{A}_t$, the selection can be written

$$a_t \in \arg\min_{a \in \mathcal{A}_t \cup \{\emptyset\}} L_{\text{train}}(Z(\mathcal{M}_t \oplus a), Y)\,. \tag{14}$$

Here Z contains simulated population responses, L_{train} is the cross-fitted prediction loss after refitting the training-side predictor, and $\emptyset$ retains the current tree. Algorithm 1 proceeds as follows. First, task-side source summaries and current responses guide source–neuron proposals, with exploration. Second, a geometry solver attempts to realize these proposals through permitted extensions; infeasible changes are discarded. Third, feasible structures are simulated on training examples, reusing unchanged neuron columns. Fourth, each task's training-side cross-fitted score compares the simulated candidate with retaining the current structure. Finally, an accepted change is committed and its structural history becomes available for later pruning.

Source utility is a proposal criterion, not a replacement for physical simulation. In the spatial-task implementations, confirmation uses cross-fitted squared prediction error, with one-hot direction targets or continuous coordinates. Each fold's readout is fitted on the remaining folds. Acceptance requires a strict reduction relative to the unchanged structure; added cable length is a tie-breaker. Synchrony scores proposals and simulated candidates on the current 1,024-example training batch using four-fold grouped out-of-fold logistic log loss, with L2 coefficient 1 on standardized feature weights and an unpenalized intercept. Dependency groups are kept within folds. It accepts a strict aggregate loss reduction, with cable length used only to break exact loss ties; no fold-wise veto or confidence threshold is imposed. In this task, cross-fitting within the growth batch is distinct from the separate confirmation split used for the spatial tasks. Periodic history reviews use training-side validation data. None of these decisions uses the report-only observer. The search is not gradient descent on all branch lengths jointly. Its proposals, feasibility constraints, and finite candidate set restrict the accessible structures.

History-based pruning reevaluates prior additions using training-side information and can remove a previous structural action. Such removal changes the geometry and induced responses, rather than merely zeroing a decoder weight. No observer fitting or evaluation sample chooses the accepted growth/pruning action. The independently refitted observer described below measures a frozen structure after these decisions.

A.6 Simulation workload and parallel execution

An ordinary proposal changes one receiving neuron. For a fixed trial set, unchanged response columns remain valid and are reused. Distinct changed-neuron configurations are simulated as independent cells across trials, packed into compatible GPU batches. Content matching is required before cached responses can substitute for simulation; changing geometry, contacts, events, or runtime content invalidates that match. Sequential structural updates are separated from parallel candidate evaluations.

Table 2 applies the same full-history accounting rule to all three tasks. We count the current population's responses and the changed-neuron response needed for each feasible candidate, before reuse of identical or cached responses. For these single-neuron proposals, $C_t = U_t$ counts feasible candidate configurations in round t before content deduplication:

$$\mathcal{W}_{\text{eval}} = \sum_t \left(N n_{0,t} + C_t n_{c,t}\right). \tag{15}$$

A neuron–trial is one neuronal response to one input realization. Each feasible candidate contributes its changed-neuron response, with unchanged population columns shared across candidates. The completed-round records supply the candidate counts; the task configuration supplies the trial counts. This measures the cumulative evaluation demand of the recorded search. Cache reuse reduces the simulations executed to satisfy that demand. Observer evaluation, remapping, and pruning

Table 2: Cumulative evaluation workload from complete training histories, with one accounting rule across tasks: current-state plus feasible-candidate neuron–trials before reuse of identical or cached responses. All rows describe HH search trajectories.

Task	Completed rounds	Feasible candidates	Evaluation neuron–trials
Synchrony	169	16,963	20,296,344
Direction	96	9,871	4,493,056
Displacement	184	20,959	9,133,824

add further work. The histories in the table have their own recorded lengths; primary performance figures use the separate nominal-48 endpoint protocol.

For synchrony, the checkpoint retains all 169 completed rounds and 16,963 feasible candidates. Each current or candidate configuration uses 1,032 trials (1,024 task examples and eight blank probes), so Eq. 15 gives $(169 \times 16 + 16{,}963) \times 1{,}032 = 20{,}296{,}344$ neuron–trials. For direction and displacement, each round uses 1,280 current-state trials per neuron and 256 trials per candidate. The same rule gives $96 \times 16 \times 1{,}280 + 9{,}871 \times 256 = 4{,}493{,}056$ and $184 \times 16 \times 1{,}280 + 20{,}959 \times 256 = 9{,}133{,}824$. Both totals match the requested-response counts in their per-round backend logs. Those logs also record 1,792 and 14,848 cache-served neuron–trials, leaving 4,491,264 and 9,118,720 executed simulations, respectively. The single-round runtime measurements below independently quantify elapsed execution time.

Task-specific configuration of the simulation pipeline. The candidate-evaluation pipeline uses HelioX (Lu et al., 2026) with device-side event delivery, task-specific batching, and memory reuse. Table 3 compares baseline and task-optimized configurations of this pipeline; both use HelioX GPU simulation. Complete rounds were replayed from frozen checkpoints with identical scientific configurations and seeds and separate cold response caches, on an eight-H100 server shared with production jobs. Direction HH and displacement Passive each used two GPUs, constant-matrix reuse, and batch packs of eight and two slices, respectively. Synchrony retained its GPU and batch settings, using direct soma-node indexing and allocation reuse without constant-matrix reuse. Times cover process launch through completion, including structural decisions and, for direction, history-based revision.

Table 3: Task-specific acceleration of the candidate-evaluation pipeline using HelioX: complete-round H100 wall times. The comparison is between two GPU pipeline configurations, not between HelioX and NEURON. Both columns use GPU simulation on the same checkpoint; all replays selected the same structural action and final geometry.

Replay	Feature values	Baseline pipeline (min)	Task-optimized pipeline (min)	Speedup
Synchrony HH	218,784	8.05 / 6.32	6.19	$1.30/1.02\times$
Synchrony Passive	224,976	6.42	5.26	$1.22\times$
Direction HH	124,160	50.52	33.33	$1.52\times$
Displacement Passive	54,784	17.40	7.09	$2.45\times$

Replays used synchrony HH round 114 (212 neuron configurations, each on 1,032 trials), synchrony Passive round 141, direction round 37, and displacement round 39. The latter two compared 970 and 428 feature columns, each containing 128 trials. Synchrony Passive, direction HH, and displacement Passive matched feature values exactly. For synchrony HH, the maximum difference between the task-optimized and baseline pipelines was 7.89×10^{-9}, equal to the difference between two baseline-pipeline replays; both baseline timings are reported to show variation under shared load. These measurements quantify the additional benefit of tailoring HelioX to the candidate-evaluation workload in Table 2.

B TASKS, OBSERVATION, AND INTERVENTIONS

B.1 EVENT-BASED TASK GENERATORS

Majority synchrony contains 32 sources: 21 informative sources and 11 distractors. Each source emits six events. A synchronous packet has span at most 1 ms, whereas asynchronous events are dispersed in time. With informative set $\mathcal{I}$ and packet indicator u_i, the target is

$$y = \mathbf{1}\left\{\sum_{i\in\mathcal{I}} u_i > |\mathcal{I}|/2\right\}. \tag{16}$$

Equal per-source event counts prevent a total-count solution. Informative-source identity defines the generator, but the target label is not supplied as an input. Separate no-distractor variants are not included in the main cohort.

The two spatial tasks contain 64 fixed sources. Spatial firing intensity changes as an activity center shifts between early and late windows. Direction detection predicts one of eight direction labels. Displacement estimation predicts two normalized displacement coordinates. Source coordinates determine event generation and geometric contact opportunities; they are not appended to Point's input features. The physical populations receive source events through their realized contacts.

Synchrony uses balanced accuracy (the mean of per-class recall); direction uses top-1 accuracy. Continuous performance averages coordinate-specific coefficients of determination:

$$\overline{R^2} = \frac{1}{2}\sum_{d\in\{x,y\}}\left[1 - \frac{\sum_n (y_{nd} - \widehat{y}_{nd})^2}{\sum_n (y_{nd} - \overline{y}_d)^2}\right]. \tag{17}$$

The mean $\overline{y}_d$ is computed from evaluation targets. Negative R^2 values are retained.

B.2 DATA SEPARATION AND THE FIXED OBSERVER

The observer fitting split contains 256 examples for synchrony and 512 for each spatial task; each evaluation split contains 1,024 examples. Within task and world, source events, target labels, and sample order are shared across HH, Passive, and Point. Recorded sample identities and source catalogs support checks for overlap between representation-training, observer-fitting, and observer-evaluation data.

The observer does not train the physical representation or Point hidden features. It refits a readout on each frozen response matrix and then evaluates that readout on held-out observer examples. The same rule applies after remapping. The authors inspected observer results during research; the data partition ensures independence from automated structural and feature-learning decisions. Complementarity uses the same held-out examples, with decoder selection confined to fitting data.

B.3 TASK-SPECIFIC LINEAR READOUTS

Readout capacity and fitting rules are shared across implementations within each task. The general objective is

$$\|Y - ZW - \mathbf{1}\mathbf{b}^\top\|_F^2 + \lambda\|W\|_F^2, \tag{18}$$

The following task-specific preprocessing and regularization define its implementation. For synchrony, encode labels as $-1, +1$, and center fitting features and targets: $Z_c = Z - \mathbf{1}\overline{\mathbf{z}}^\top$ and $\mathbf{y}_c = \mathbf{y} - \overline{y}\mathbf{1}$. Then

$$\lambda = 0.05\max\left(\mathrm{tr}(Z_c^\top Z_c)/16, 10^{-10}\right), \tag{19}$$

$$\widehat{\mathbf{w}} = (Z_c^\top Z_c + \lambda I)^{-1} Z_c^\top \mathbf{y}_c, \qquad \widehat{b} = \overline{y} - \overline{\mathbf{z}}^\top \widehat{\mathbf{w}}. \tag{20}$$

Scores at least zero predict the positive class. The scale floor precedes multiplication by 0.05. The comparison uses this refitted ridge decoder on frozen Point features, not the auxiliary logistic head used during Point training.

For both spatial tasks, standardize features with fitting-set means and population standard deviations; replace standard deviations below 10^{-9} with one. If $\widetilde{Z}$ denotes these features and $D = [\widetilde{Z}, \mathbf{1}]$, then

$$\widehat{\Theta} = (D^\top D + 4\,\mathrm{diag}(1, \ldots, 1, 0))^{-1} D^\top Y. \tag{21}$$

These equations are implemented as linear solves. Direction uses eight-dimensional one-hot targets and predicts the largest output. Continuous estimation uses two coordinates. Intercepts are unpenalized. Including intercepts, readouts contain 17, 136, and 34 coefficients for synchrony, direction, and displacement. Appendix E specifies the separately cross-validated decoder used to estimate C.

B.4 Point comparison

Point has one hidden layer of 16 tanh units. For source feature vector $\mathbf{f}_i$, its feature and source–unit budget are

$$z_m = \tanh\left(\sum_i \mathbf{w}_{mi}^\top \mathbf{f}_i + \beta_m\right), \qquad K = \sum_{i,m} \mathbf{1}\{\|\mathbf{w}_{mi}\|_2 > 0\}. \tag{22}$$

For synchrony, $\mathbf{f}_i$ is the source's packet-span synchrony indicator. This gives Point a timing primitive that the physical implementations must extract from event-driven dynamics and their response transform. For the spatial tasks, $\mathbf{f}_i$ contains early and late event counts. One source–unit group therefore contains one or two coefficients. Biases and the auxiliary training head add parameters outside K; physical B_{eq} and Point K are access measures, not equal parameter counts.

Point trains on examples from the first four recorded physical learning rounds in chronological order, without performance selection: 4,096 synchrony examples (proposal data) or 5,120 spatial-task examples (discovery plus confirmation data). Input standardization is fitted on these training examples and then frozen. This uses shared recorded data but does not equalize cumulative training exposure with iterative physical search.

The hidden layer and an auxiliary head use Adam (Kingma & Ba, 2015) for 2,000 fixed full-batch steps, learning rate 0.01, and L2 coefficient 0.001 on feature weights and head parameters. Losses are binary cross-entropy, multiclass cross-entropy, and mean squared error. After each update, only the K source–unit groups with largest Euclidean norm are retained. There is no observer-dependent early stopping. Hidden features are frozen before the shared observer readout is fitted. Each evaluated integer budget uses eight optimization seeds per world.

B.5 Remapping fixed physical structures

The strong-contact graph is bipartite, connecting external sources to receiving neurons. Its edges have aggregate contact gain at least 0.9 (numerical tolerance 10^{-12}), using the same distributed-contact rule as the physical model. Source-held remapping applies degree-preserving two-edge switches: it retains the participating source identities and every source and neuron degree. Full-graph remapping changes participating source identities as well, while preserving neuron degrees and the distribution of source degrees. Neither intervention grows a new tree.

Within each neuron, a bijection assigns source identities to existing contact sites. Unchanged strong assignments are retained where possible; other sites remain unchanged when compatible with this bijection. Sites, gains, membrane settings, and morphology are fixed. The bijection also preserves source-wise gain aggregation within each neuron and hence B_{eq}. What changes is the event stream arriving at a site. These counterfactual assignments isolate input allocation at fixed morphology and may lie outside the configurations reachable by geometric growth.

Each selected state has eight independently generated remaps per family. We simulate each condition on identical fitting/evaluation examples and refit its readout. For either performance P or population gain C, we subtract the within-family remap median from the learned value, then average these paired effects equally over worlds. The independent replication unit is the world, with remaps nested within its learned state. Full-graph remapping jointly perturbs selection and assignment; we interpret each family through its own learned-minus-remapped effect.

B.6 REPLICATION AND BUDGET SELECTION

Each physical implementation has four training seeds in each of four worlds, giving 96 trajectories across the three tasks. Point has eight optimization seeds per world and integer budget. States are selected by proximity to budget anchors, not by score. Missing measurements are not interpolated. Seeds are averaged within each world before the four worlds receive equal weight. Bands report the full range of world means, not confidence intervals. Remapping uses one HH trajectory per world; its eight realizations per family are paired interventions rather than eight independent training replicates.

C DENDRITIC DEVELOPMENT THROUGH A COMPLETE TRAINING TRAJECTORY

Figures 5 and 6 follow the direction-detection HH population in world 914120, training seed index 0, from initialization to its primary nominal-48 endpoint (round 51, $B_{\mathrm{eq}} = 48.0634$). Four neurons are followed with consistent colors. Population snapshots sample 12 stages; individual rows show eight stages each. Compact curves retain every recorded round. This illustrates Algorithm 1, complementing the four-world statistics in the main text.

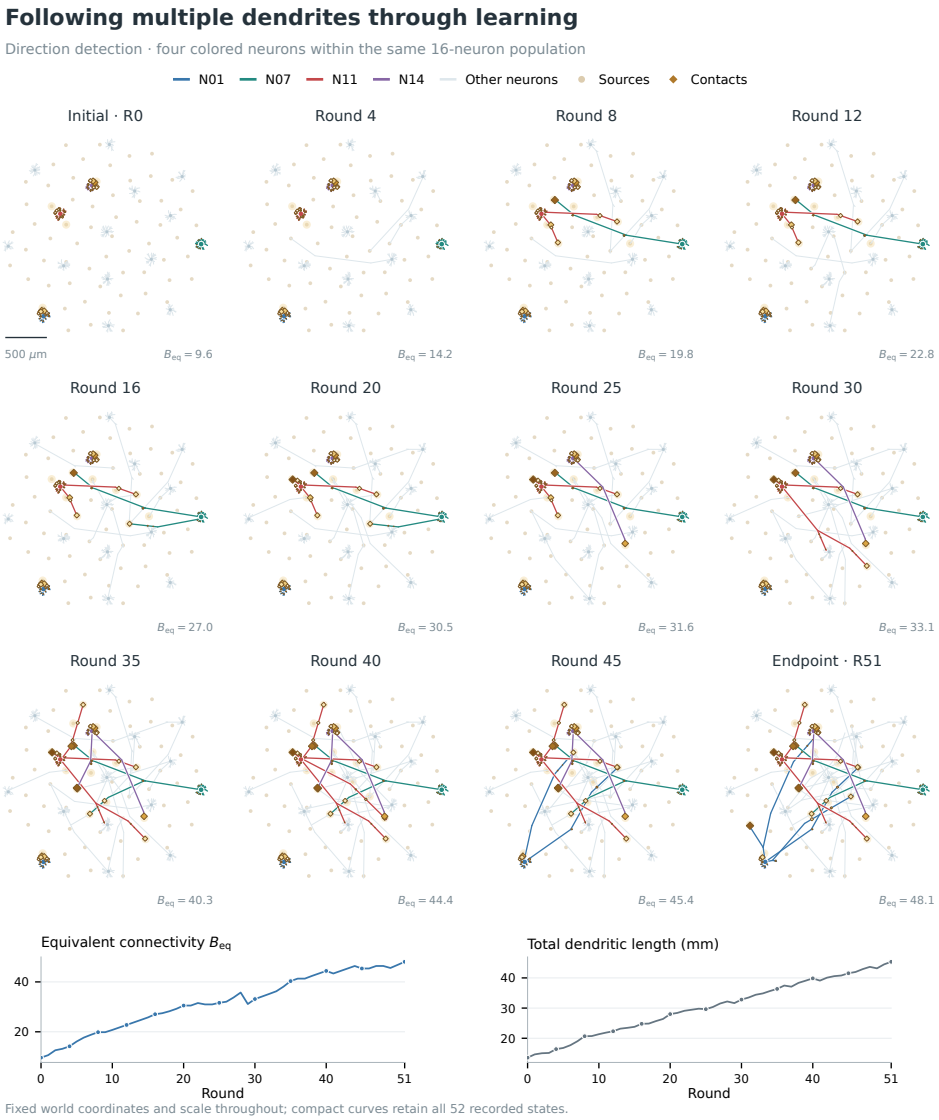

Figure 5: **Multiple dendrites develop within the same population.** Twelve recorded stages show all 16 neurons in fixed world coordinates. N01, N07, N11, and N14 retain their blue, teal, red, and purple identities; other neurons are pale. Sources are gold, and diamonds mark contacts on the highlighted cells. Equivalent connectivity and total cable length are shown in compact curves below, with dots identifying the displayed stages. All geometry is reconstructed from the training checkpoints.

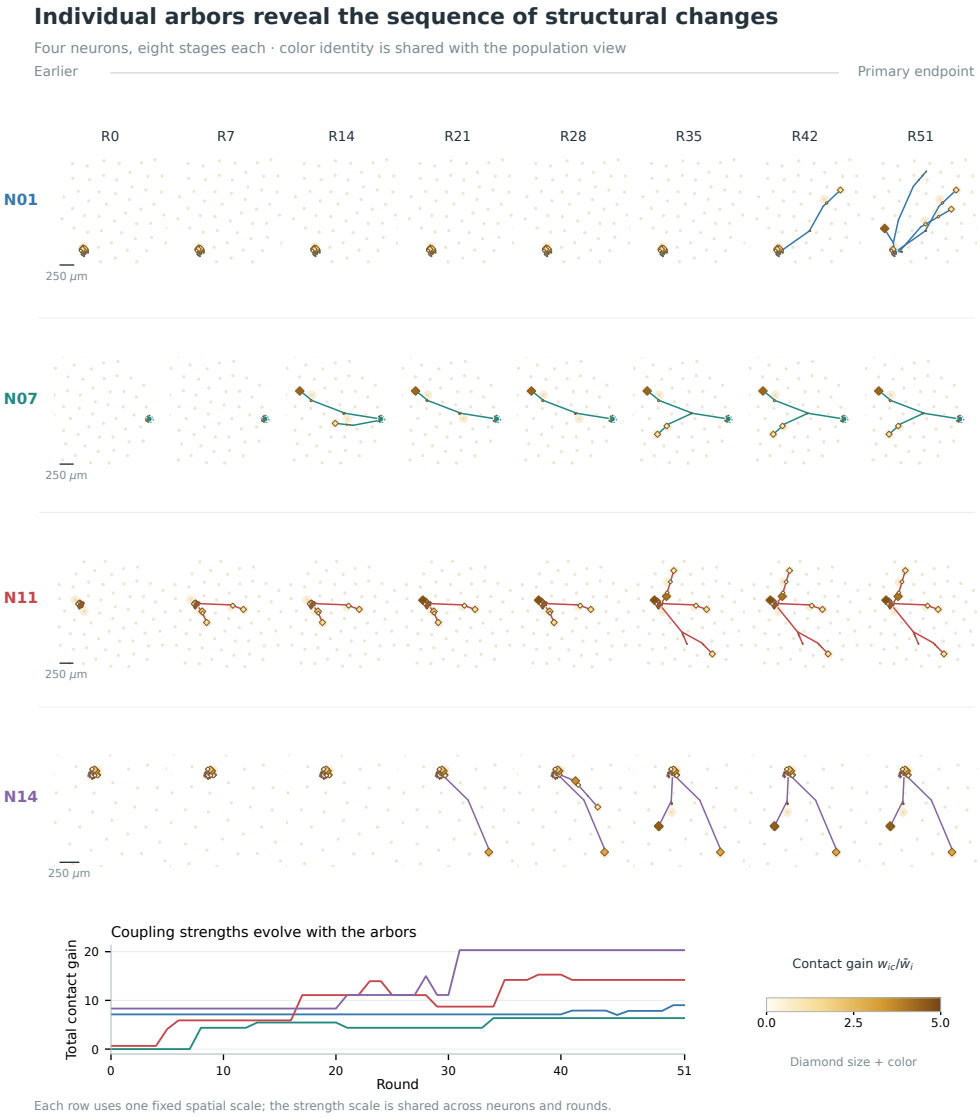

Figure 6: **Individual arbors show distinct developmental sequences.** Rows follow the same four neurons across eight stages, with a fixed spatial scale within each row (250 μm bars in the first column). The separate views reveal branch extensions and revisions without overlap from other cells. Diamond size and color encode merged contact gain $w_{ic}/\bar{w}_i$ on one shared scale. The small curves sum these gains over all contacts of each neuron at every recorded round; values are not capped at one. Source halos indicate coupling.

The contact reconstruction uses the exact configuration and active-terminal sampling and aggregation rules in Appendix A. Endpoint connectivity and cable length agree with the independent audit to numerical precision. Neurons were chosen to show spatially separated arbors and different timings of structural change. These are illustrative views of one trajectory; the original four-world performance and representation statistics are unchanged. All geometry snapshots, configuration hashes, and reconstruction inputs are preserved with the figure sources.

D REPLICATION, BUDGETS, AND NUMERICAL RESULTS

D.1 COHORT AND STATISTICAL UNITS

Every main aggregate includes the original valid world and three additional worlds generated with matched task-specific initialization rules. Four seeds in each world for each of two physical implementations and three tasks give 96 trajectories. Initial morphology and observer data are paired within task/world; training seeds change learning streams. Initial physical evaluations are shared across the four seeds of a world and implementation and are not four independent initial samples.

Physical seeds are averaged within world, then the four world means receive equal weight. Point uses eight seeds per world/budget and the same outer averaging. The nominal-48 Point export contains 3,824 fitted models across the covered integer budgets. Figure bands are the full range of world means, not standard errors or confidence intervals. With only four independent worlds, even a two-sided exact paired sign-flip test cannot attain $p < 0.05$ (its minimum is $2/2^4 = 0.125$). We therefore report magnitudes and consistency without claiming conventional significance from these comparisons.

The remapping cohort uses one HH trajectory per task/world and eight remaps per family/state. It is not the full four-seed physical comparison cohort. The four worlds, rather than neurons, remaps, or multiple budget states from one trajectory, are the independent units of its paired summaries.

D.2 ACTUAL BUDGETS AND COVERAGE

True initial physical budgets average 7.70 for synchrony and 10.47 for the spatial tasks. Figure 4 includes these states. Intermediate comparison anchors use nearby observed geometries, with a maximum deviation of 2.1 budget units in the performance-curve assembly. Ties are resolved by archive order and geometry identity, not score. When a complete physical cohort has no admissible measurements at an anchor, the aggregate point is omitted. In particular, sparse archived original-world synchrony HH histories do not support every requested intermediate anchor. No interpolation fabricates missing observations, and plotted horizontal coordinates are measured budgets averaged over the selected states.

For the original historical runs, endpoint extraction selects archived states closest to 48. For the new runs, it uses the frozen nominal-48 endpoint records; discrete additions and monitoring can produce overshoot. The actual endpoint budgets in Table 4 make this distinction visible. These are budget-selected endpoints, not each run's best observed score. The primary budget range was set during the study after progress inspection, rather than preregistered. The primary results exclude the optional higher-budget extensions.

Figure 3 displays six nominal anchors: $B_0 + 2$, 16, 24, 32, 40, and 48. Each family is paired to the same learned geometry. Selected historical remapping states can deviate from the nominal anchor; their actual budgets are retained. Additional early states are available for the three new worlds, and Appendix E shows all per-world measurements rather than averaging an incomplete set into a four-world curve. Negative effects are retained.

Table 4: Nominal-48 endpoints. Scores and budgets are averaged within world and then over four worlds; brackets give the range of world mean scores. Accuracy is shown as a percentage and displacement as R^2. Physical B_{eq} and Point K are different access measures.

Task	Model	Actual budget	Score	World-mean range
Synchrony	HH	48.41	80.38	[79.44, 82.15]
Synchrony	Passive	48.91	79.16	[77.12, 81.81]
Synchrony	Point	48.00	94.01	[91.85, 95.63]
Direction	HH	48.27	98.69	[98.29, 98.97]
Direction	Passive	48.45	99.20	[98.83, 99.54]
Direction	Point	48.00	99.53	[99.33, 99.67]
Displacement	HH	48.27	0.860	[0.851, 0.881]
Displacement	Passive	48.74	0.884	[0.880, 0.886]
Displacement	Point	48.00	0.909	[0.906, 0.911]

Table 5: Endpoint remapping effects. Each effect is learned minus the median of eight remaps, then averaged over four worlds. Accuracy losses are percentage points; displacement loss is in R^2 units. Population-gain differences are dimensionless.

Task	Family	ΔP	ΔC	Worlds with $\Delta C > 0$
Synchrony	Source-held	4.09	0.056	4/4
Synchrony	Full-graph	11.08	0.190	4/4
Direction	Source-held	4.88	0.068	4/4
Direction	Full-graph	11.69	0.120	4/4
Displacement	Source-held	0.170	0.138	4/4
Displacement	Full-graph	0.190	0.158	4/4

Table 6: World-level endpoint intervention effects. W0 is the original illustrated world; W1–W3 are the three new worlds in fixed generation order. ΔP is reported in native score units (accuracy fractions or R^2). Each row uses its own frozen geometry and eight remaps per family.

Task	World	B_{eq}	ΔP_{SH}	ΔP_{FG}	ΔC_{SH}	ΔC_{FG}
Synchrony	W0	47.75	0.063	0.120	0.100	0.222
Synchrony	W1	48.39	0.035	0.101	0.031	0.160
Synchrony	W2	48.01	0.041	0.123	0.060	0.219
Synchrony	W3	48.85	0.025	0.099	0.034	0.160
Direction	W0	48.24	0.038	0.108	0.056	0.105
Direction	W1	48.06	0.059	0.124	0.075	0.128
Direction	W2	48.25	0.039	0.105	0.067	0.122
Direction	W3	48.23	0.060	0.130	0.073	0.125
Displacement	W0	47.91	0.187	0.200	0.120	0.164
Displacement	W1	49.43	0.150	0.172	0.111	0.124
Displacement	W2	48.55	0.187	0.212	0.167	0.187
Displacement	W3	48.12	0.156	0.175	0.153	0.156

E Population representation and figure-selection details

E.1 Decoder-based population gain

$$C = \frac{L_{\text{single}} - L_{\text{population}}}{L_{\text{null}}}, \qquad \Delta C_f = C_{\text{learned}} - \text{median}_r\, C_{f,r}. \tag{23}$$

The paired effect compares learned gain with the median gain over remaps in family f. For each frozen learned or remapped response matrix, we estimate population and single-neuron predictive losses independently of the fixed reporting decoder. Synchrony targets are scalar $0/1$ values, direction targets are one-hot vectors, and displacement targets are two continuous coordinates. For evaluation examples $(\mathbf{z}_n, \mathbf{y}_n)$, loss is

$$L(\widehat{f}) = \frac{1}{n_{\text{eval}}} \sum_n \|\mathbf{y}_n - \widehat{f}(\mathbf{z}_n)\|_2^2, \qquad L_{\text{null}} = \frac{1}{n_{\text{eval}}} \sum_n \|\mathbf{y}_n - \overline{\mathbf{y}}_{\text{fit}}\|_2^2. \tag{24}$$

The same null is used for all conditions of a task/world because targets are shared. $C = (L_{\text{single}} - L_{\text{population}})/L_{\text{null}}$ is positive when the population decoder outperforms the selected single-neuron decoder. It is not constrained to $[0, 1]$. A negative value is possible when combining features generalizes less well than using one neuron. A positive learned-minus-remapped ΔC means a larger population advantage in the learned condition, not necessarily a larger absolute population score for every condition separately.

Decoder selection uses five-fold cross-validation on fitting data only, with a fixed random seed (93601), stratified by class for classification and unstratified for regression. Within each training fold, features are standardized using that fold's means and population standard deviations, replacing standard deviations at most 10^{-12} with one. We solve a ridge objective with *mean* squared loss and unpenalized intercept. The regularization grid is

$$\lambda \in \{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}. \tag{25}$$

Fold errors are weighted by their sample counts. For the population, we choose the minimum-error regularization. For each neuron, we separately choose regularization and then select the neuron with smallest fitting cross-validation error. All choices are repeated for every remap. Chosen decoders are refitted on all fitting examples and evaluated once on the corresponding held-out responses. No evaluation loss selects the single neuron or regularization.

This definition captures decoder-accessible complementarity relative to the best fitting-selected single neuron. It can reflect noise averaging and distributed target information as well as distinct tuning. The estimand is predictive improvement under the specified linear decoder. Performance loss and ΔC characterize accuracy and population advantage on the same held-out samples.

E.2 Morphology and tuning examples

Figure 2 keeps the original world as a shared visual reference across tasks and shows round 22. All 16 arbors are drawn; N02 is highlighted in red, with a small initial-arbor inset. Background arbors use a pale color and white outlines so contacts remain readable. Source halos represent coupling and contact markers remain visible over branches. The voltage panels use the highlighted neuron before and after growth under the same fitting stimulus. Their vertical offsets aid comparison; the 40 mV bars preserve physical amplitude interpretation.

Figure 3 uses a direction HH trajectory from a new world at measured $B_{\text{eq}} = 48.0634$. The pair is N10/N11. Selection uses fitting responses and morphology: a tuning-quality threshold of 0.3, first-harmonic phase separation of at least $140°$, and a preference for developed branching, maximizing the smaller and then total counts of sections longer than $70\,\mu$m. Fitting preferred phases are approximately $210.83°$ and $52.48°$, separated by $158.35°$. This fitting-based selection provides a readable example of developed arbors with distinct tuning. Population-level conclusions use the four-world quantitative analysis.

Direction tuning curves plot evaluation-trial means and SEM for each direction. The learned heatmap orders neurons by fitting-derived preferred phase and uses the raw late-minus-early spike-count feature on a fixed scale of ± 5 spikes. It does not normalize each neuron to equal peak amplitude. Figure 7 adds the first stored source-held and full-graph realizations using the same order and scale; these examples were not selected by the size of the remapping effect.

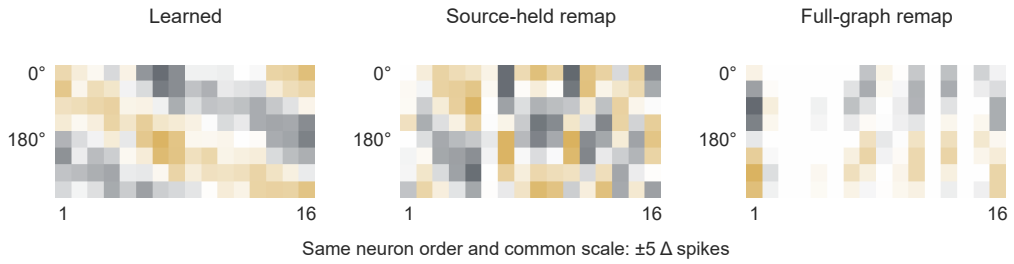

Figure 7: **Matched direction response matrices.** Learned responses and the first stored remap of each family share one fitting-derived neuron order and one absolute color scale. The matrix examples visualize how fixed physical populations respond to changed input identities; the quantitative analyses summarize all eight remaps per family and four worlds.

E.3 Full measured budget trajectories

The compact main panels use common nominal anchors for legibility, with actual measured budgets on the horizontal axis. Figure 8 displays the available per-world effect trajectories, including the denser early measurements in the new worlds. This preserves early effects near zero or below zero and exposes unequal archived state density. Lines connect observations to guide reading; they do not create additional simulated states. World-level endpoint effects are listed in Table 6.

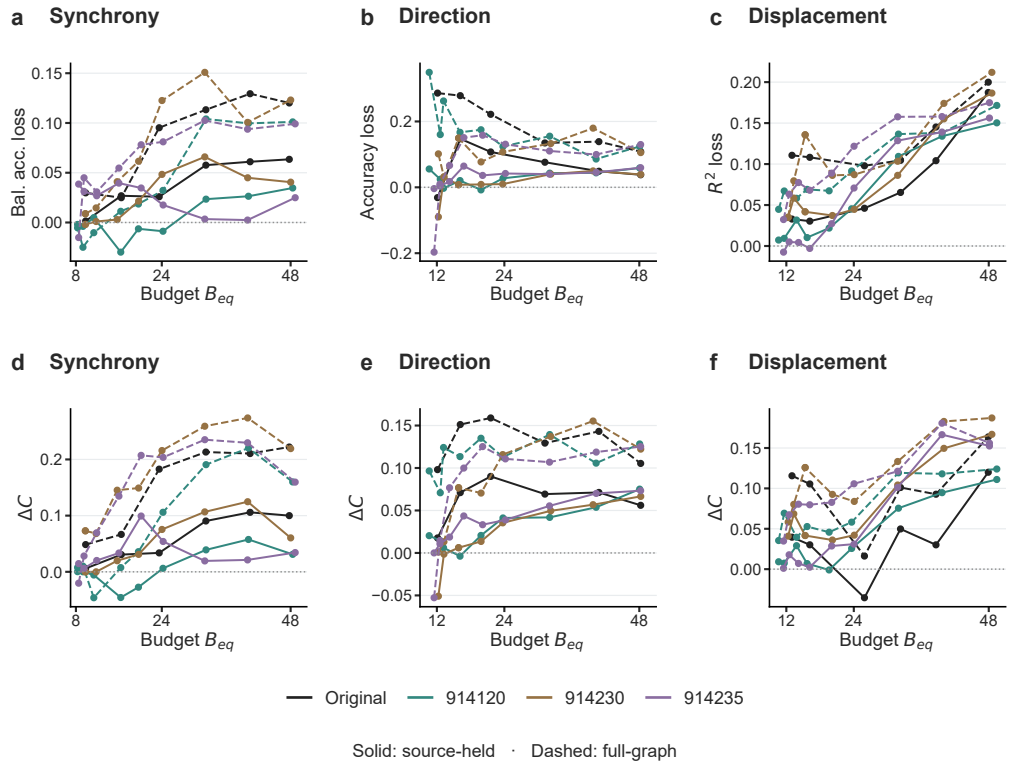

Figure 8: **Complete available early-to-endpoint remapping effects.** Curves show each world's learned-minus-median-remapped performance and decoder-based population gain. All available states are retained, including extra early states where archived responses permit them. The original world is included throughout; the main figure summarizes all four worlds at its selected anchors.