The Physical Effects of Learning

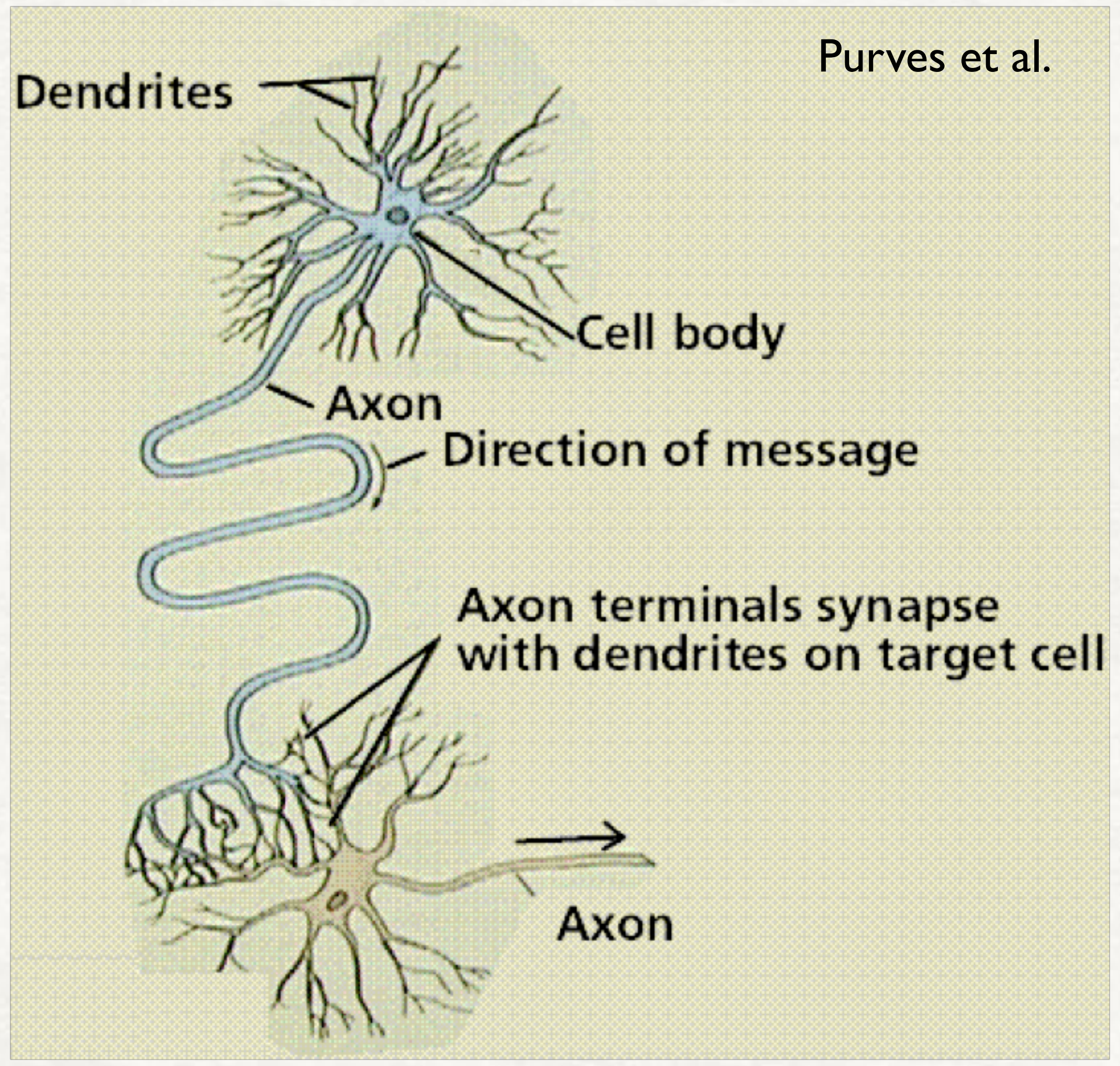
Vijay Balasubramanian

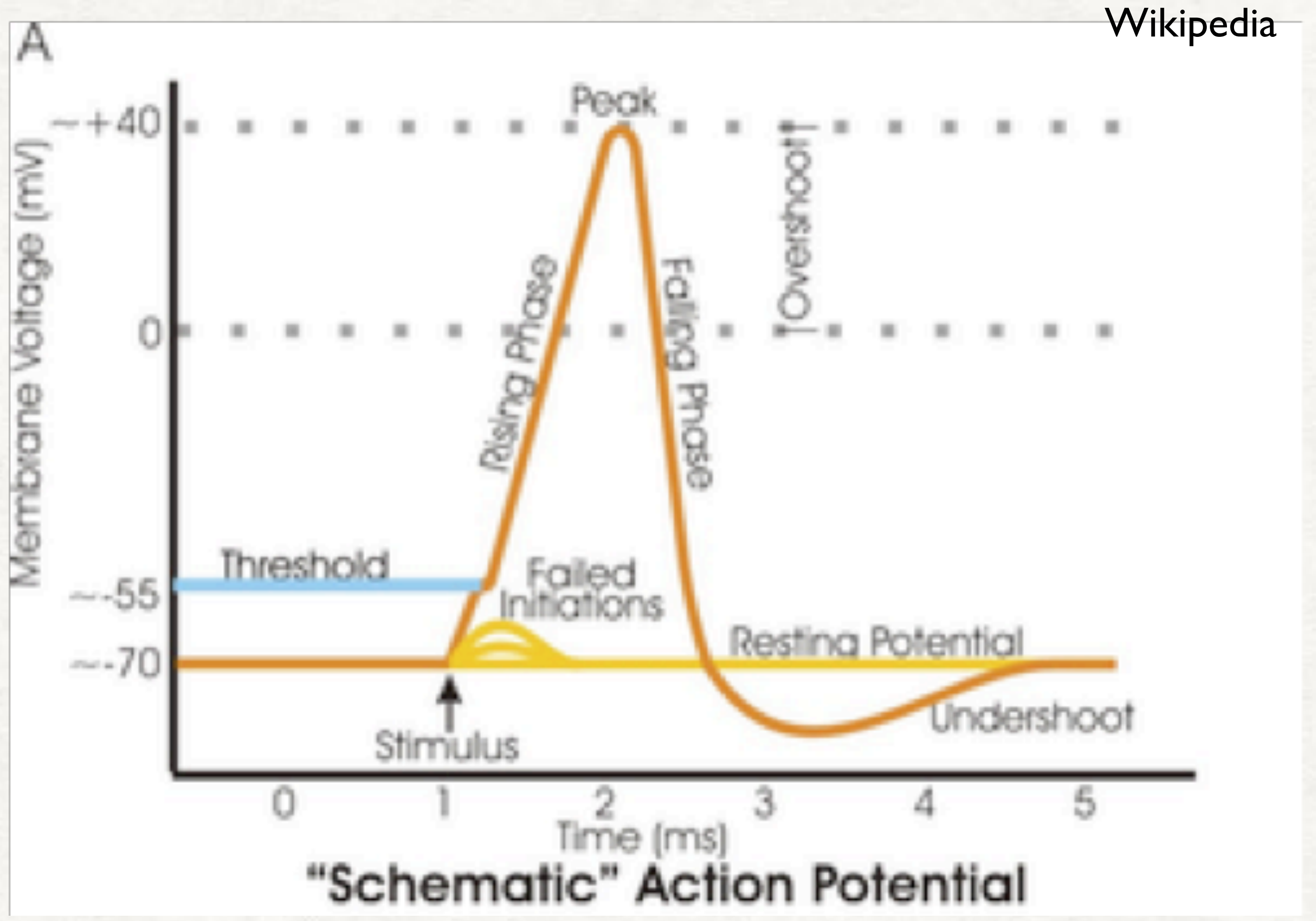
University of Pennsylvania

Based on work with **Menachem Stern & Andrea Liu**

January 12, 2024: Aspen Center for Physics

A paradigmatic physical learning network

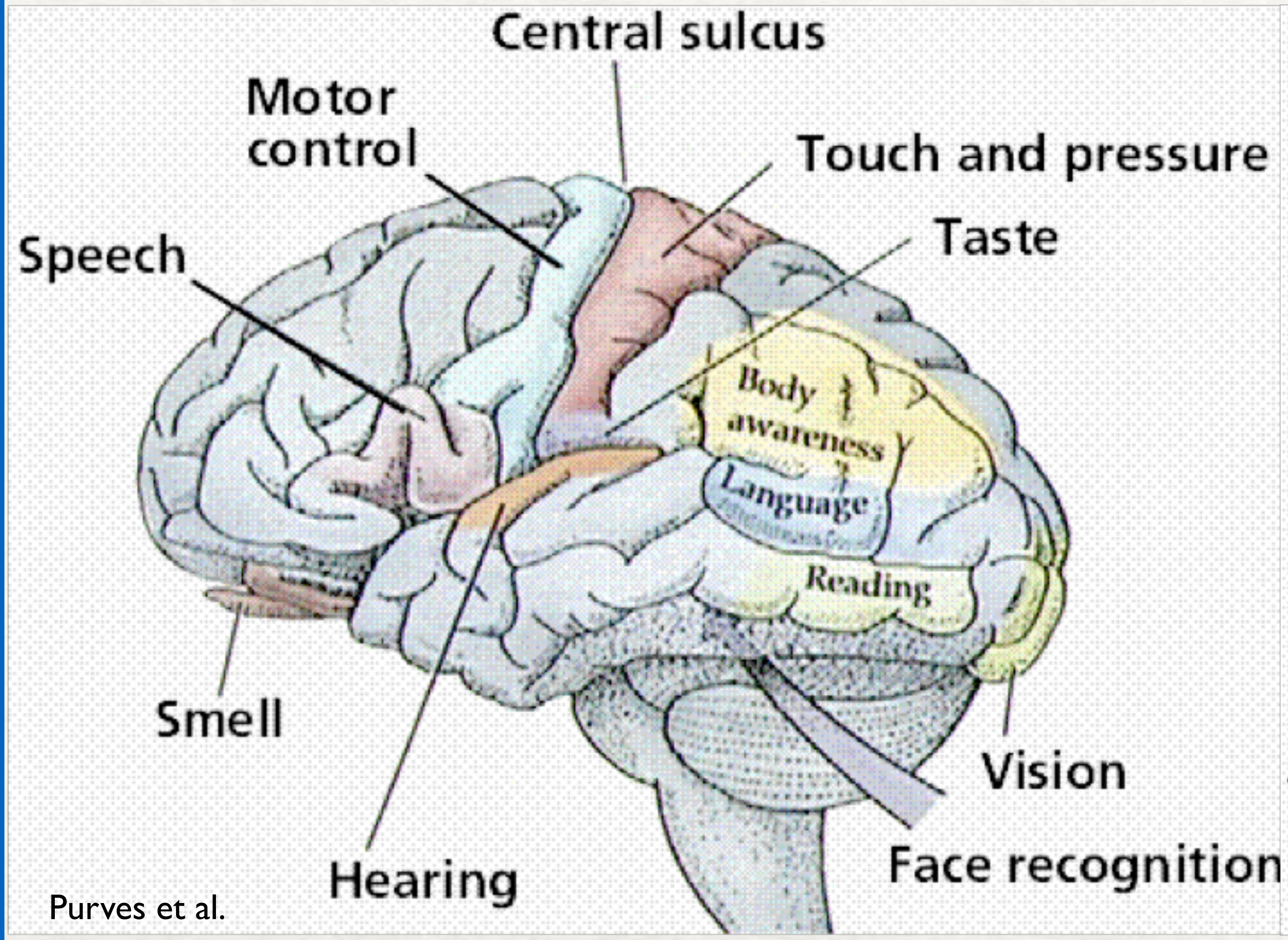


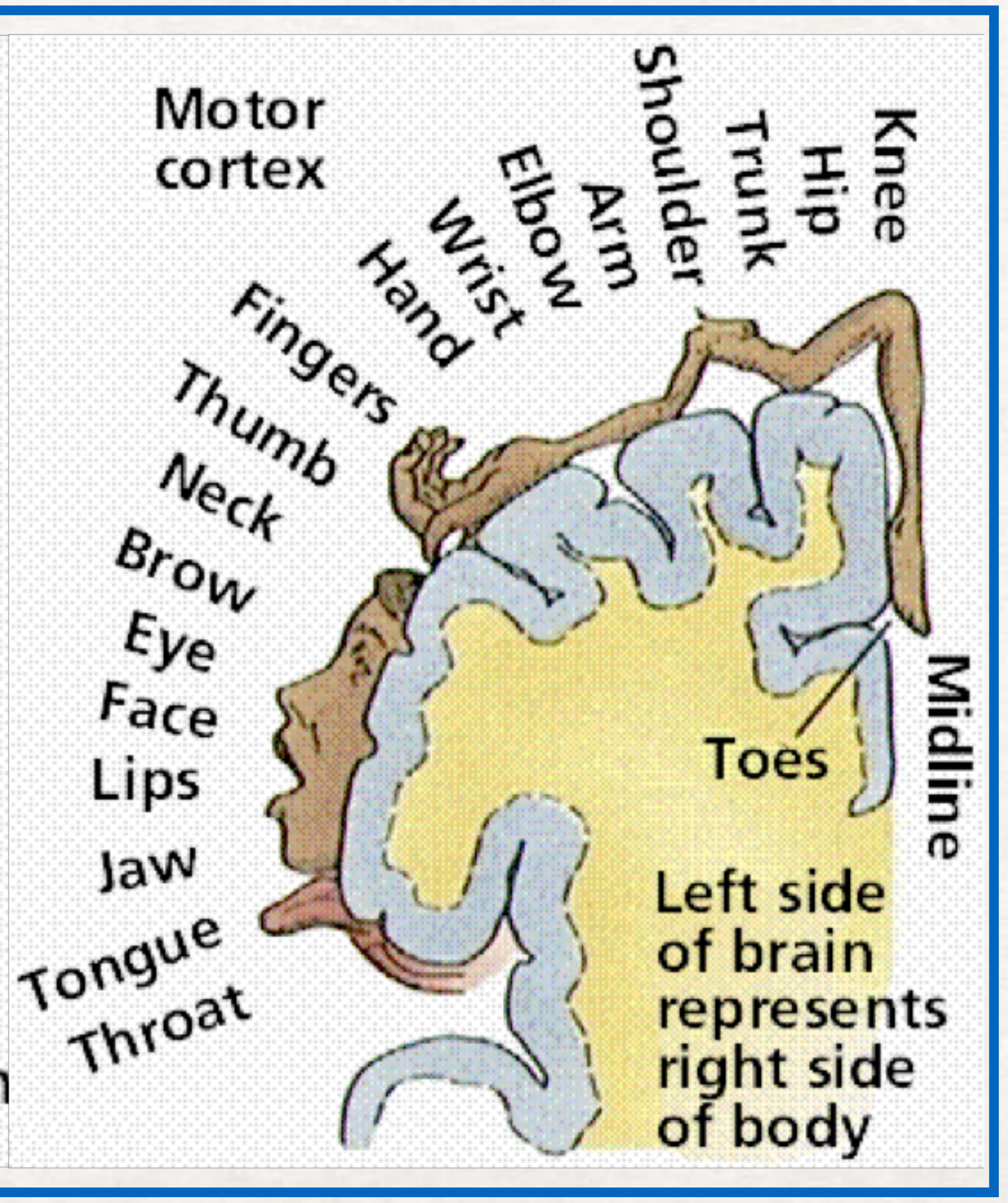


Neurons are electrical circuit elements.

Brain is 2% of body weight, but 20% of metabolic load. 10x the cost of muscle.

Brain Power. PNAS 2021. VB





Computational function is distributed across areas of the brain and also within each area

Brain areas are organized in interacting networks

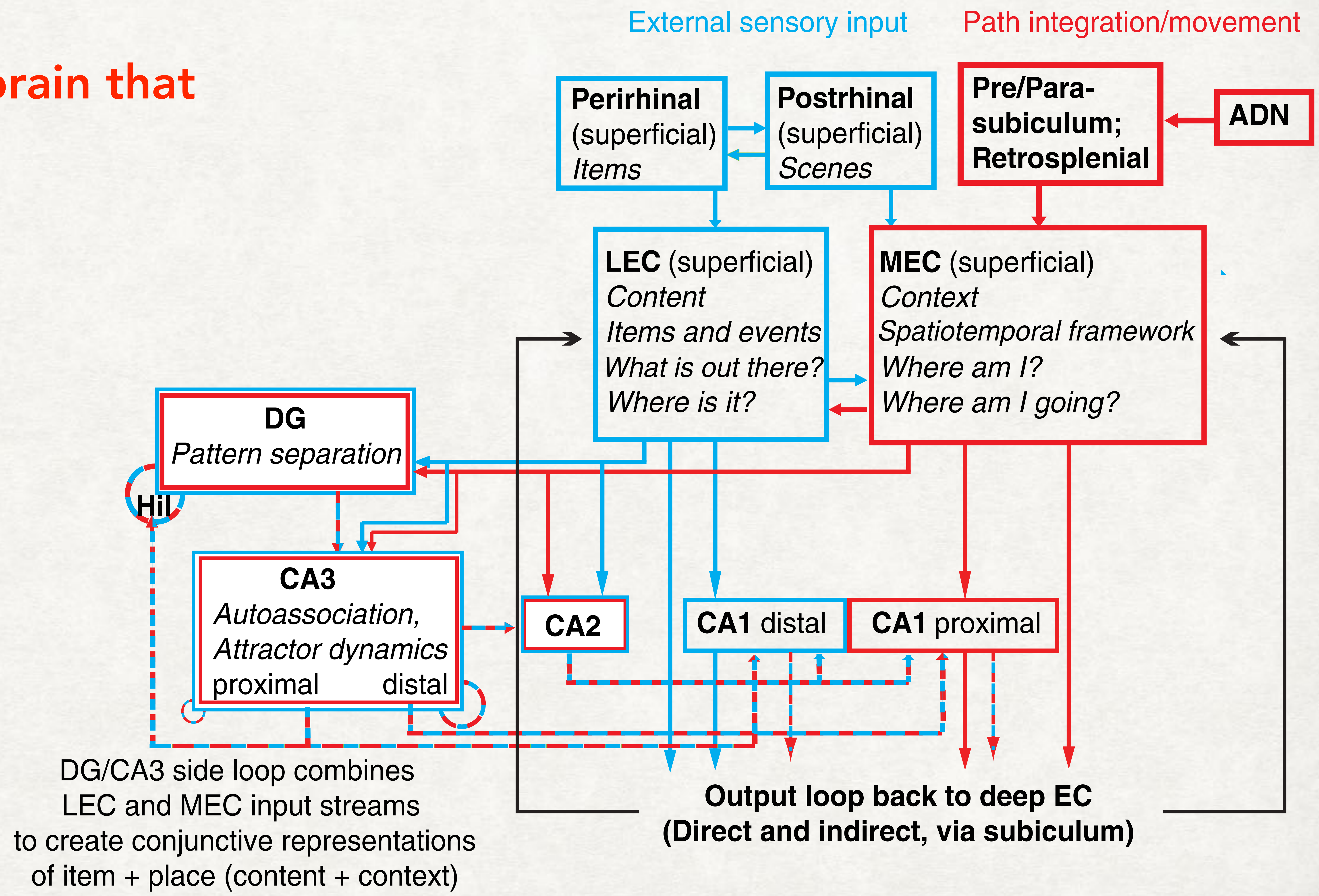
Information flow between areas of brain that act cooperatively

Sensory input: Perirhinal and Postrhinal cortex

Movement input: Pre- and para-subiculum; Retrosplenial cortex

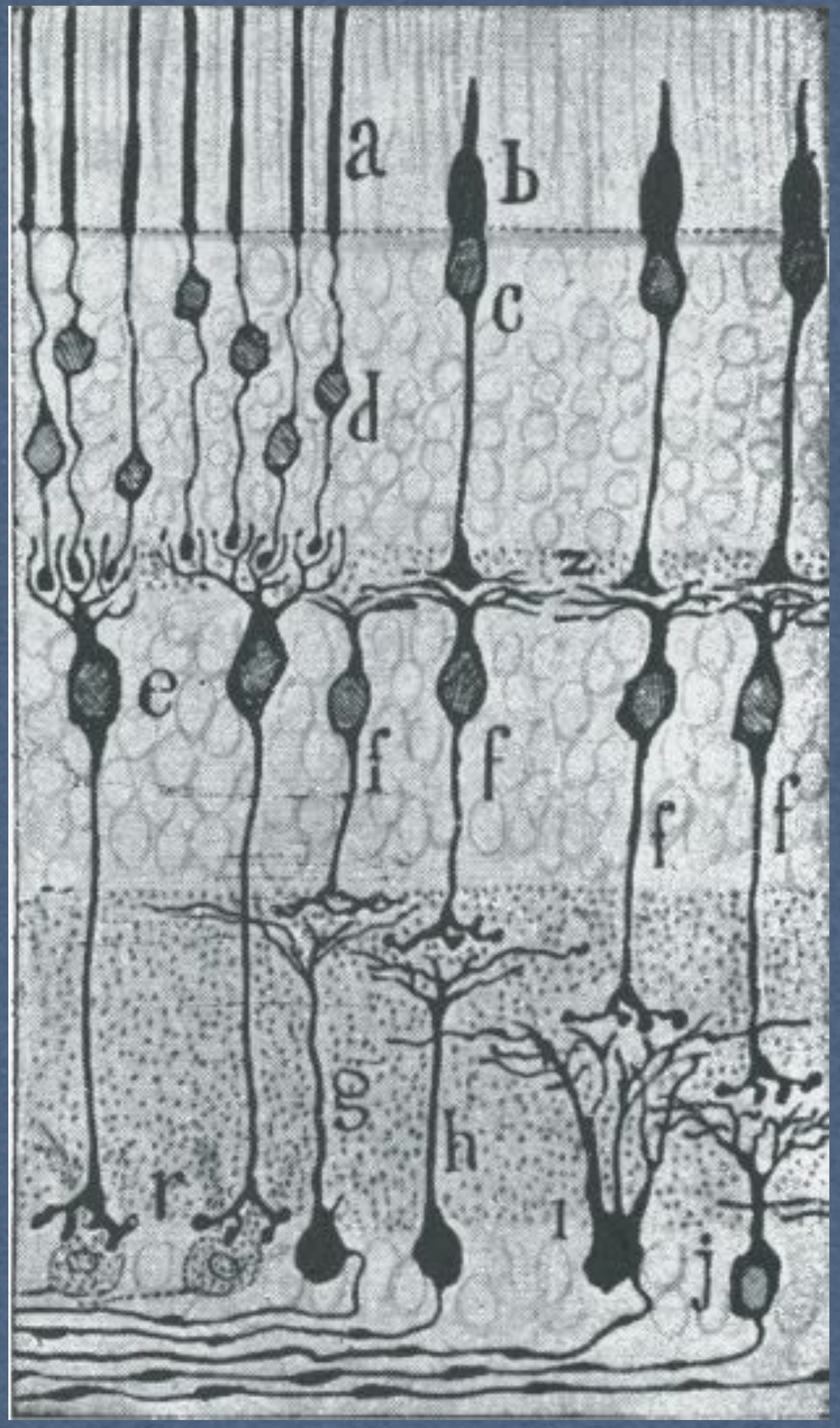
Location and navigation: Entorhinal cortex & Hippocampus

Cell Types: border cells, boundary vector cells, head direction cells, speed cells, place cells, landmark cells, object vector cells, egocentric cells, allocentric cells

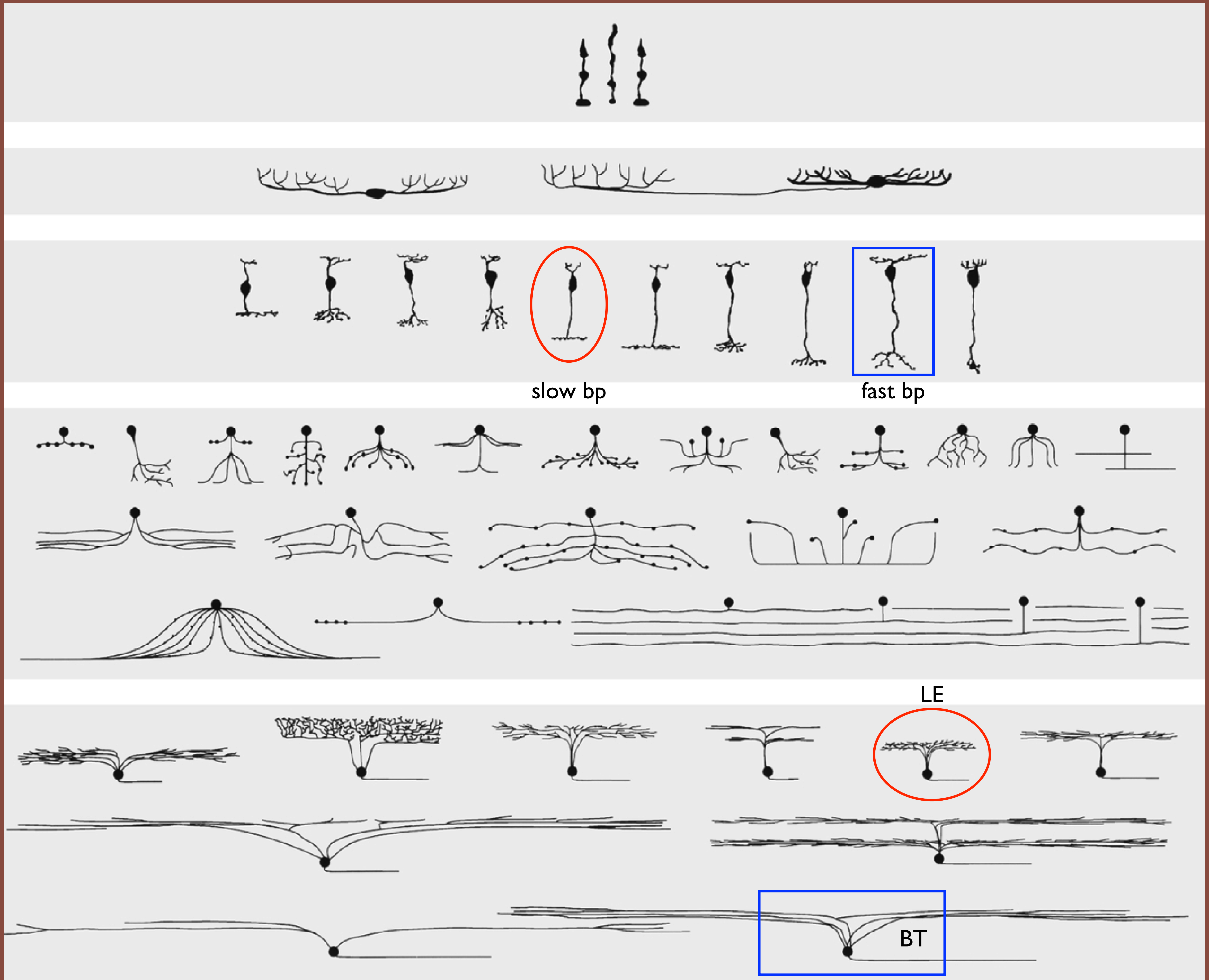


(Knieirim, Current Biology)

Microcircuits are distributed and heterogeneous



Retina: Ramón y Cajal 1917

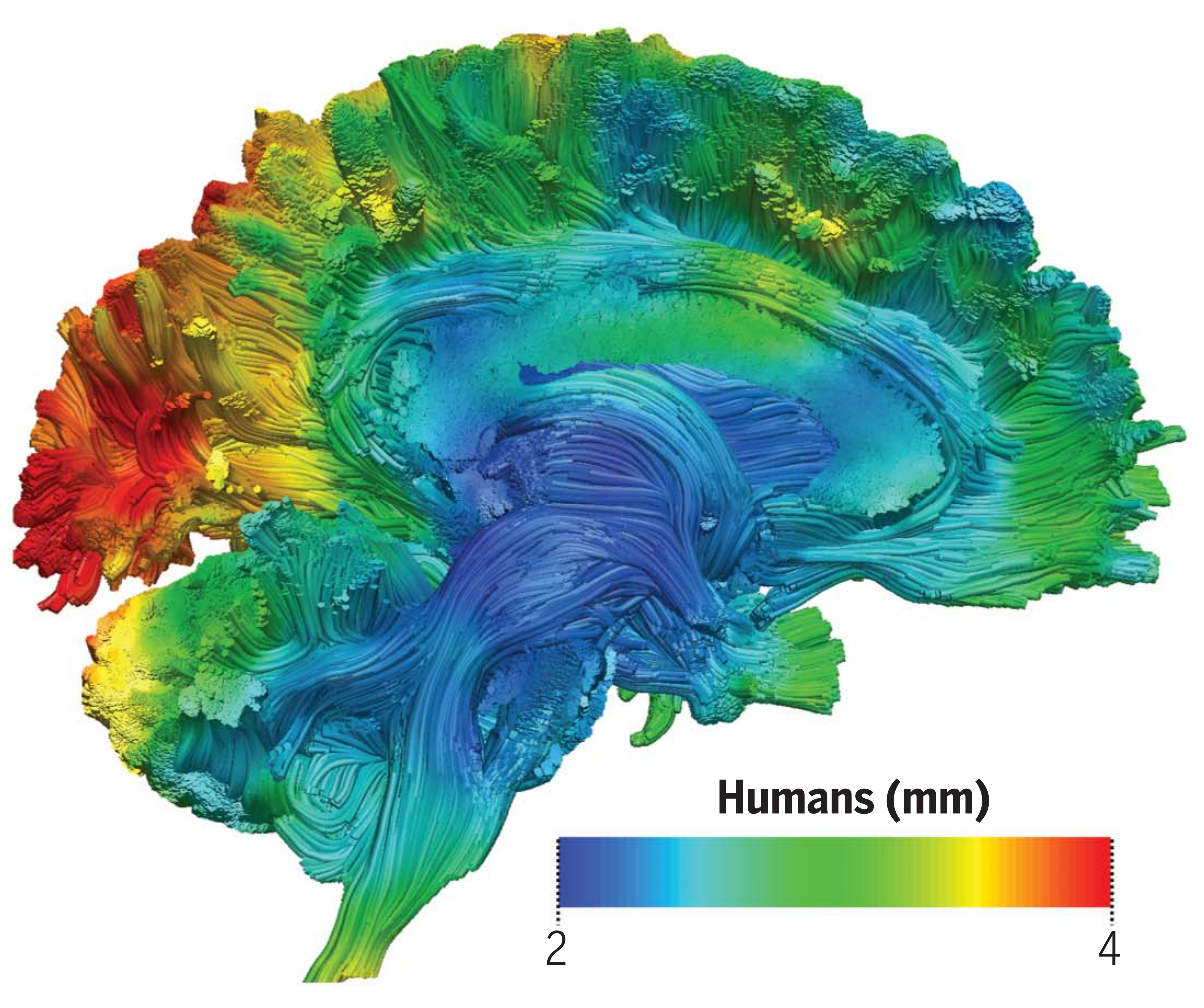


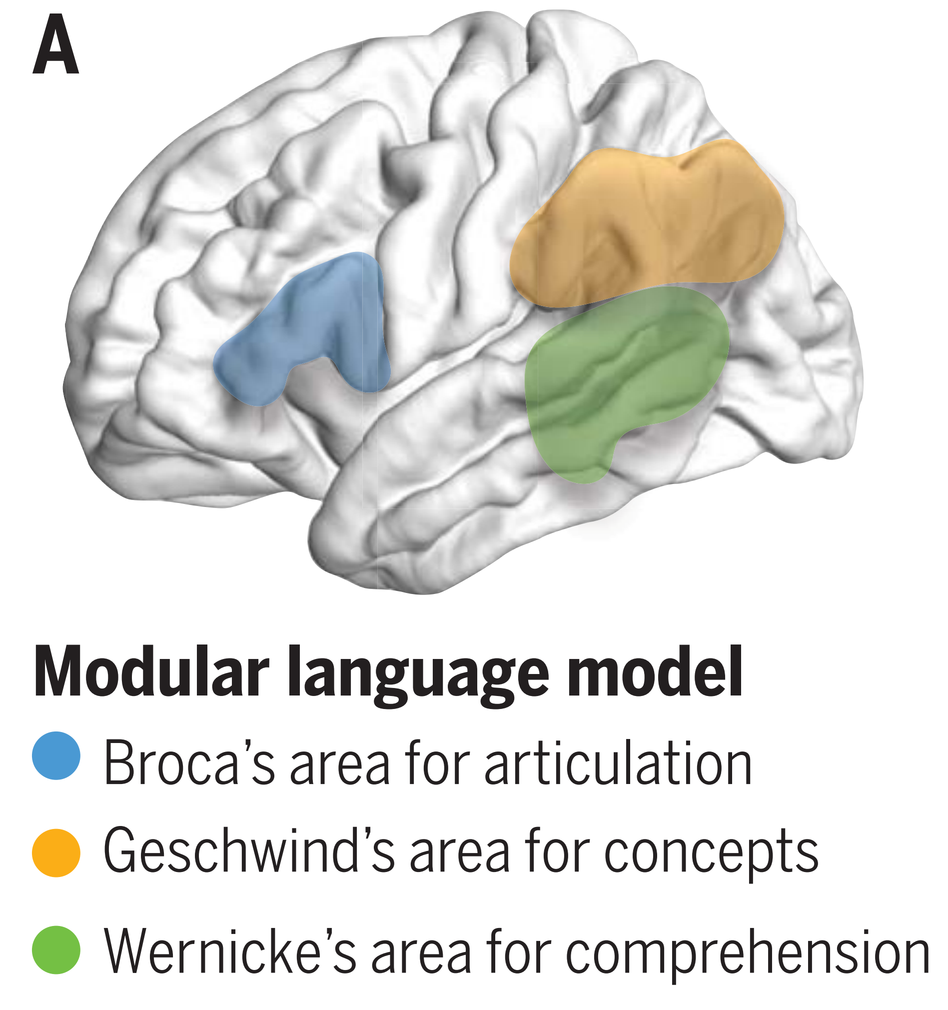
3 cones
2 horizontal cells
10 bipolars
30 interneurons
15 ganglion cells

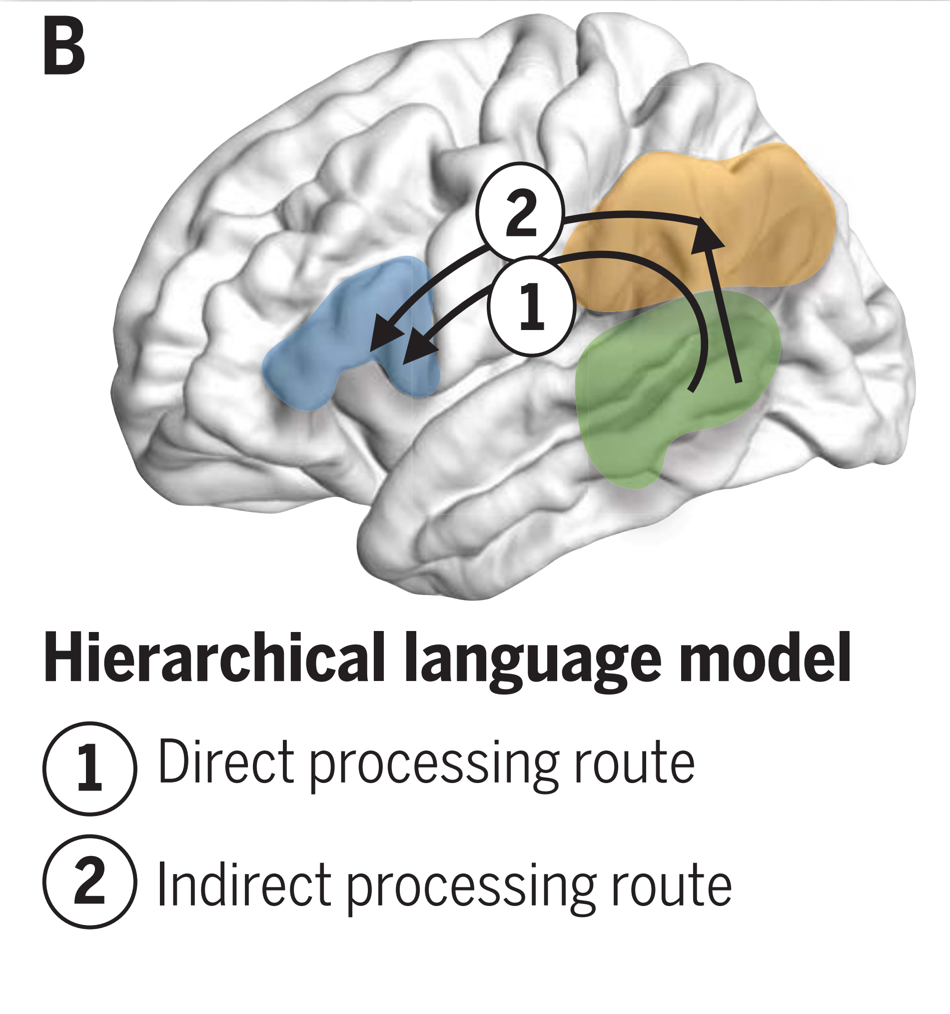
Masland 2001

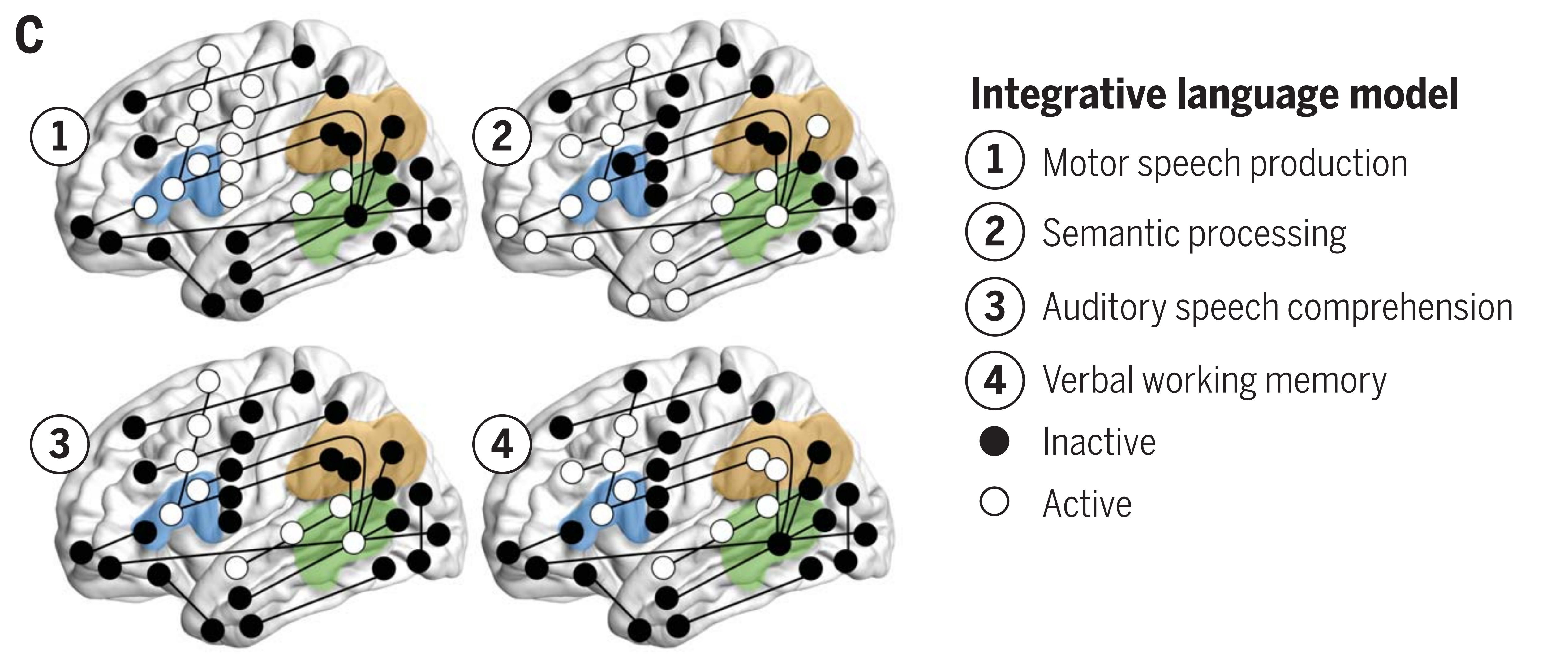
Structural heterogeneity supports computational and resource efficiency in the brain (VB, Proc. of IEEE, 2015)

Brain networks reorganize actively and flexibly









Images:
de Schotten
and
Forkel,
Science,
2022

- Brain regions are extensively connected by nerve tracts
- These can be imaged and measured
- The effective connectivity changes depending on the task

Function emerges from effective patterns of connectivity that reorganize with each task.

Learning and circuit reorganization via local rules

Neurons learn by autonomously rewiring their own circuits.

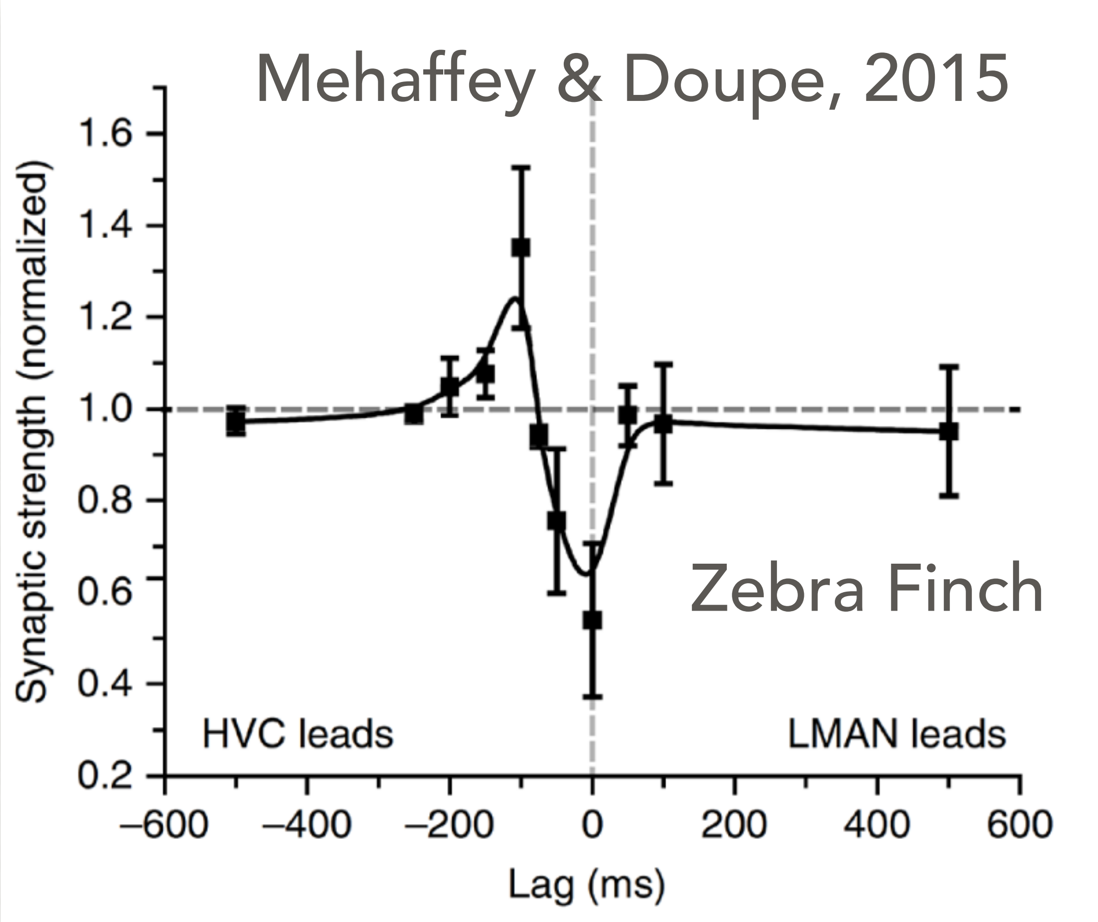
One mechanism is Spike Timing Dependent Plasticity (STDP). For example:

- Suppose two connected neurons fire voltage spikes
- If the first neuron fires before the second one, the synapse strengthens
- If the first neuron fires *after* the second one, the synapse weakens.

You can also use neuromodulators as global knobs to modulate learning

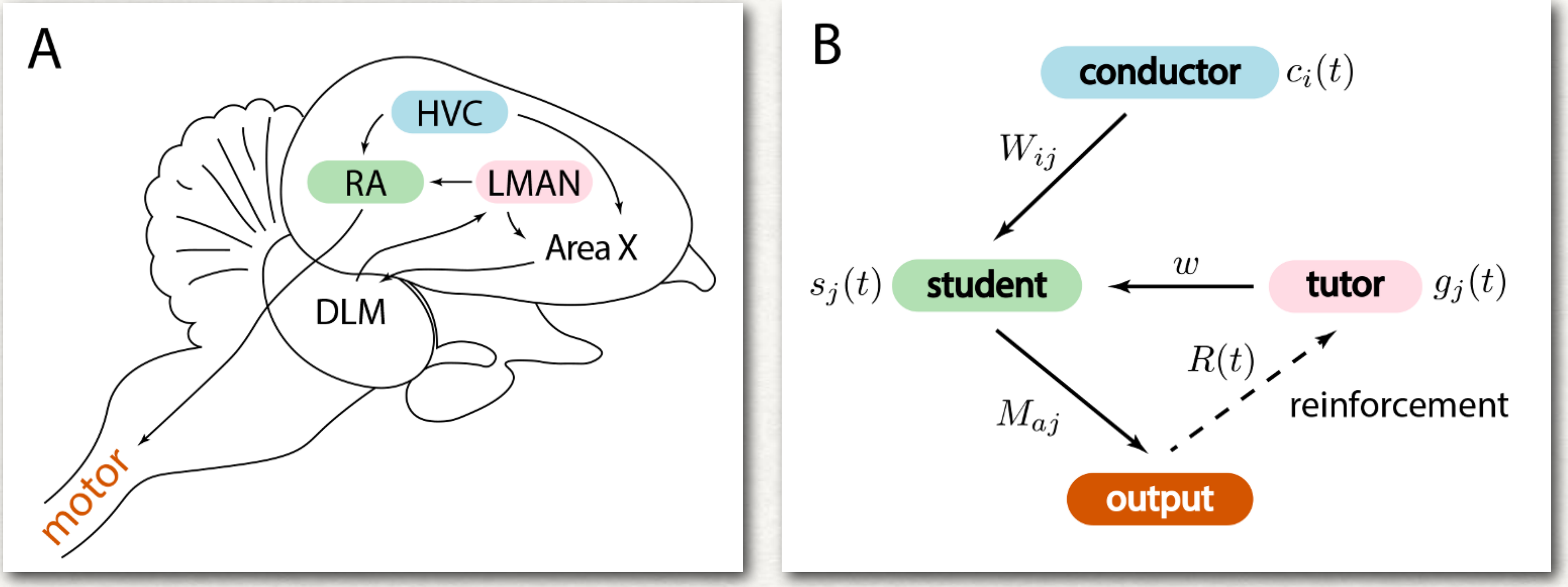
- Dopamine released on a synapse reinforces changes that lead to surprising rewards
- Norepinephrine release allows emotion to affect synaptic plasticity

An individual neuron or synapse has no direct knowledge of how other neurons are adjusting themselves.



Learning requires whole brain cooperation: e.g., song learning

Major brain areas of the **songbird** act collectively during song learning



The **tutor** should match the teaching style (reinforcement protocol) to **student** learning style (synaptic plasticity rule) for efficient learning

- **Student**: Neurons in RA control the muscles to produce sounds. It must learn a sequence of muscular movements.
- **Conductor**: Neurons in HVC produce a sequence of patterns marking time. After learning each pattern drives RA to pull and push the right muscles at each moment
- **Tutor:** LMAN drives exploration by injecting variability while providing guidance based on comparison of the desired song to the actual output.

Physical effects of learning in the brain

You become what you learn

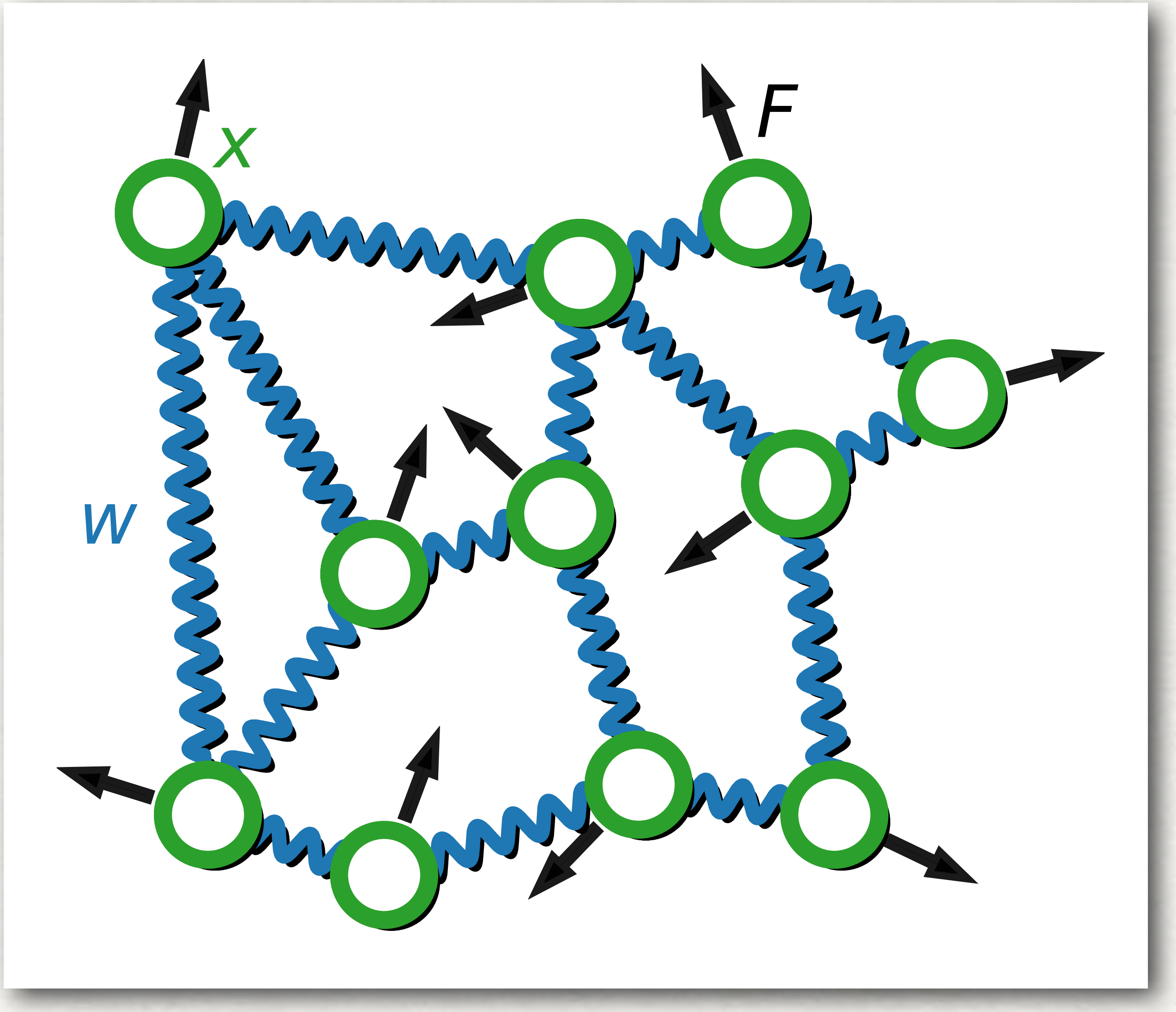
1. Synaptic connectivity reorganizes with learning
2. Low dimensional response repertoires (response patterns and dynamics)
3. Response repertoires aligned with the relevant features of relevant inputs and desired outputs

Question: How do local learning rules at synapses produce these effects?

A Strategy

Construct a pared-down network with minimal features of real neural networks and see if it can learn, and whether it also develops the structural adaptations seen in the brain.

A class of tractable physical networks



Near equilibrium the network has energy:

$$E(\vec{x},\vec{w}) \approx E(\vec{x}^0,\vec{w}) + \frac{1}{2}(\vec{x}-\vec{x}^0)^T H(\vec{w})(\vec{x}-\vec{x}^0)$$

- $x_a,\ a = 1 \cdots N$ are physical degrees of freedom (nodes)
- $w_i,\ i = 1 \cdots N_w$ are learning degrees of freedom (edges)
- $\overrightarrow{F}$ are (weak) forces applied to the nodes

Physical Hessian: $$H_{ab} = \frac{1}{2}\sum_i \phi_i(w_i)[L_{ai}R_{ib} + R^T_{ai}L^T_{ib}] \equiv \sum_i H^i_{ab}$$

$\phi(w_i)$ is an edgewise nonlinearity. L and R are fixed, and specify the network geometry.

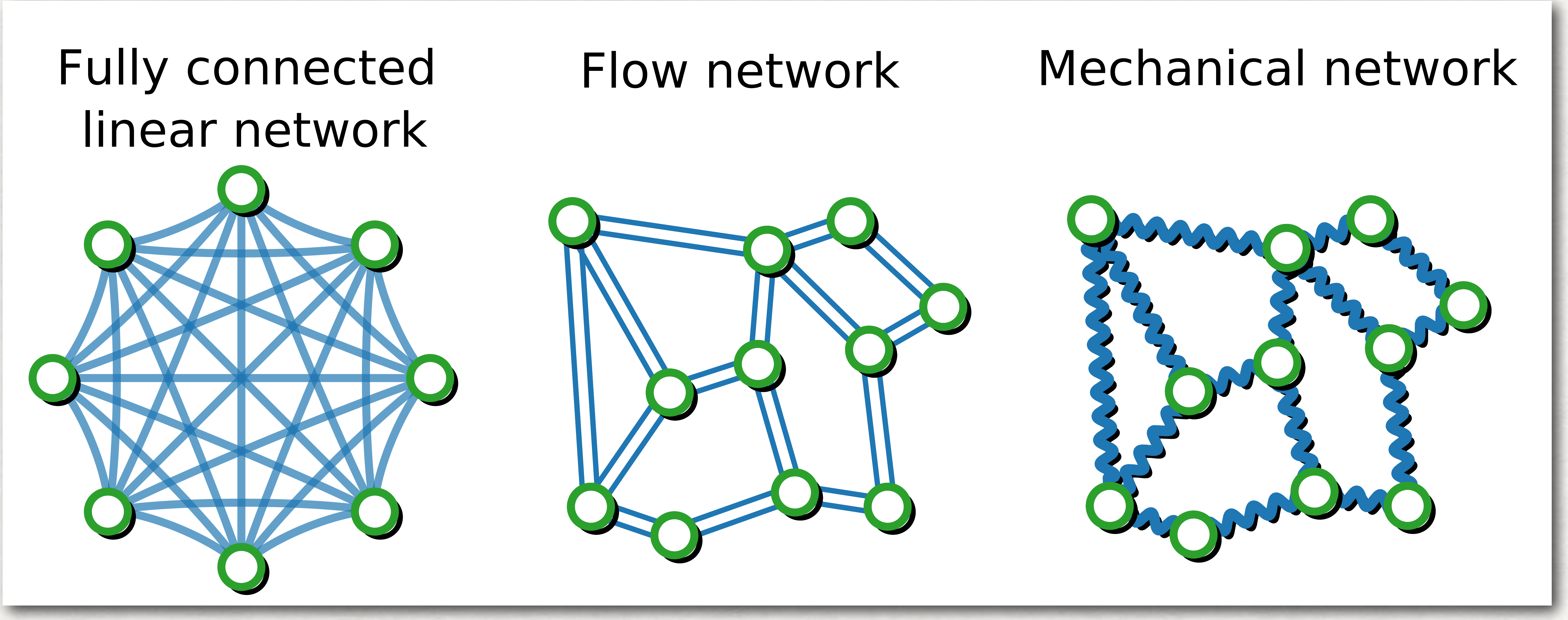
The effect of forces:

$$E^F(\vec{x},\vec{w}) = E(\vec{x},\vec{w}) - \vec{F}\cdot\vec{x}$$

$$\frac{\partial}{\partial x}E^F(\vec{x},\vec{w}) = 0 \implies \frac{\partial}{\partial x}E(\vec{x},\vec{w}) = \vec{F}$$

$$\vec{x}^F - \vec{x}^0 = H^{-1}\vec{F}$$

Simple examples: flow and elastic networks

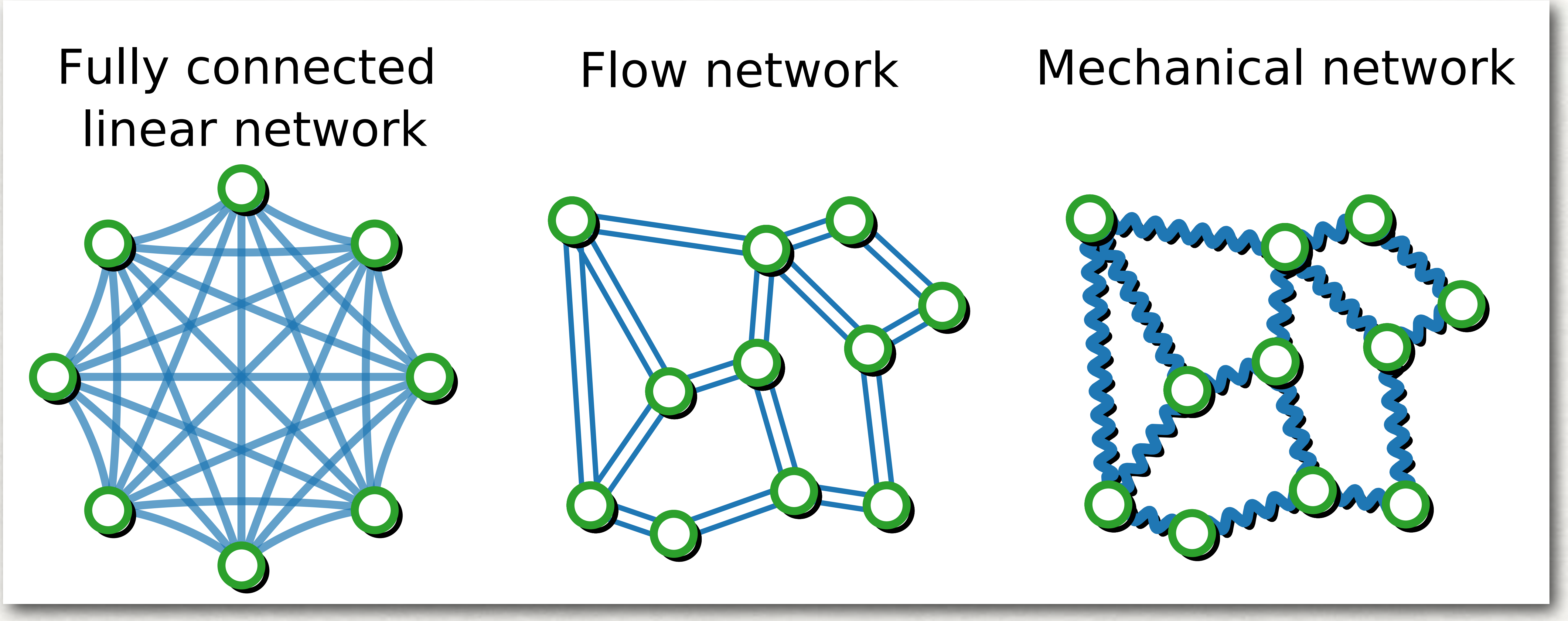


Linear Network: $E(\vec{x}, W) = \frac{1}{2}\vec{x}\, H(W)\, \vec{x}$ with $H(w) = \frac{1}{2}(W + W^T)$

Flow/Elastic Network: $E(\vec{x}, W) = (1/2)(\vec{x} - \vec{x}^0)\, H(\vec{w})\, (\vec{x} - \vec{x}^0)$

with $H(\vec{w}) = \Delta^T \operatorname{diag}(\vec{w})\, \Delta$ where Δ is an edge incidence matrix (+1 for incoming, -1 for outcoming) with arbitrarily assigned orientation of each edge

Forces, inputs and outputs



The effect of forces

$$E^F(\vec{x}, \vec{w}) = E(\vec{x}, \vec{w}) - \vec{F} \cdot \vec{x}$$

$$\frac{\partial}{\partial x} E^F(\vec{x}, \vec{w}) = 0 \quad \Longrightarrow \quad \frac{\partial}{\partial x} E(\vec{x}, \vec{w}) = \vec{F}$$

$$\vec{x}^F - \vec{x}^0 = H^{-1} \vec{F}$$

Notice that H^{-1} and hence the "free state" $\vec{x}^F$ depends non-locally on all of the edge weights: the response of depends non-locally on the edges

Inputs: Pick some input nodes. Apply inputs as forces at these nodes

Outputs: Pick some output nodes and measure their deviation

Example task: Allostery — push some set of nodes, get responses at others

A physical learning protocol

Tasks as constraints: $c(\vec{x}^F - \vec{x}^0) = 0$

Cost (a sum over tasks):

$$C(\{\vec{x}_r\}, \vec{x}^0(\vec{w})) \equiv \frac{1}{2} n_T^{-1} \sum_r [c^{(r)}(\vec{x}_r - \vec{x}^0)]^2$$

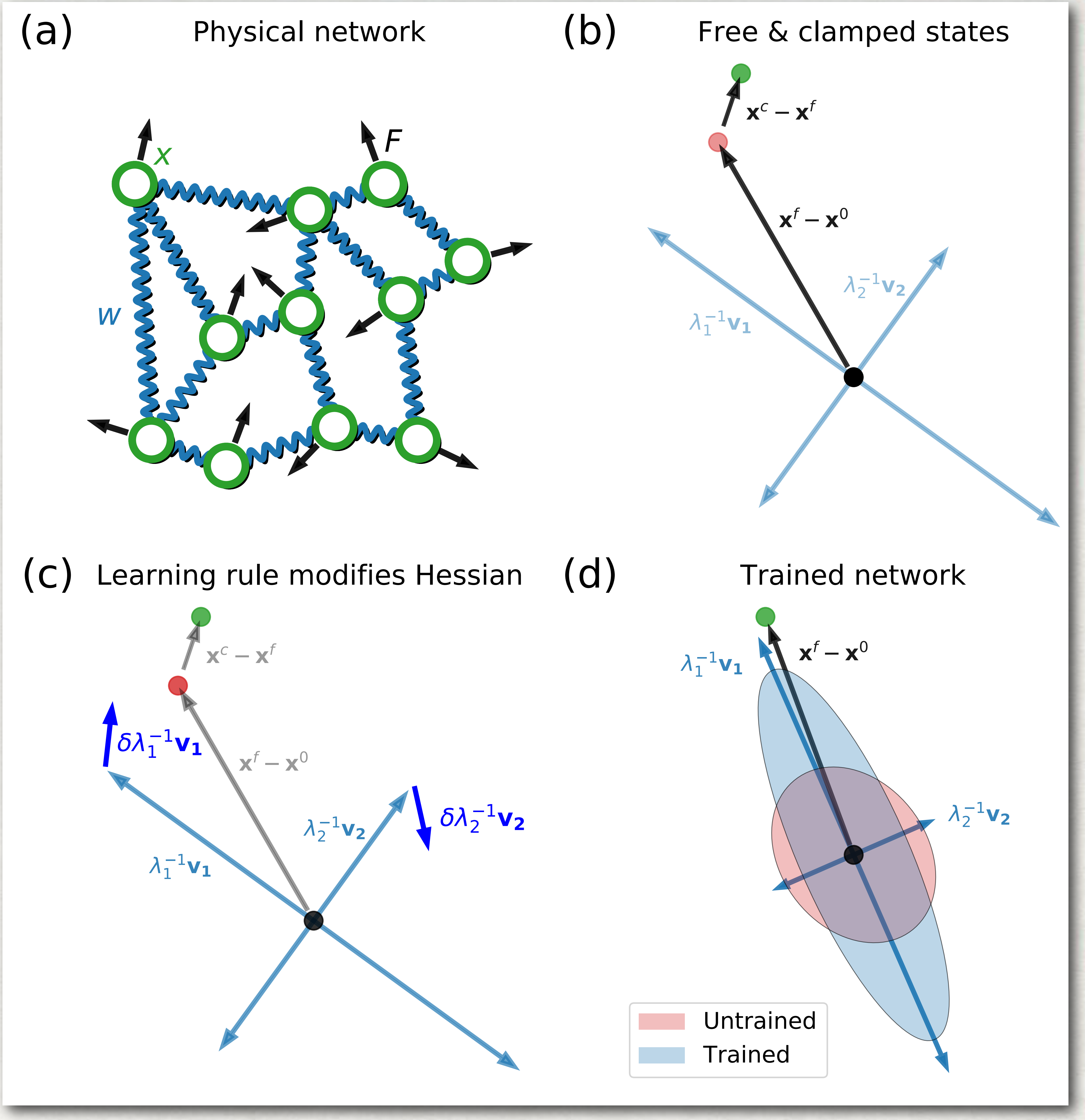
Contrastive Learning (Scellier & Bengio):

Nudge the free state by an additional output force: $\overrightarrow{F}^O = \nabla_{\vec{x}} C$. This gives a "clamped state" $\vec{x}^C - \vec{x}^F = \eta H^{-1} \overrightarrow{F}^O$ with energy $E^C = E^F + \eta\, C$

Contrastive function:

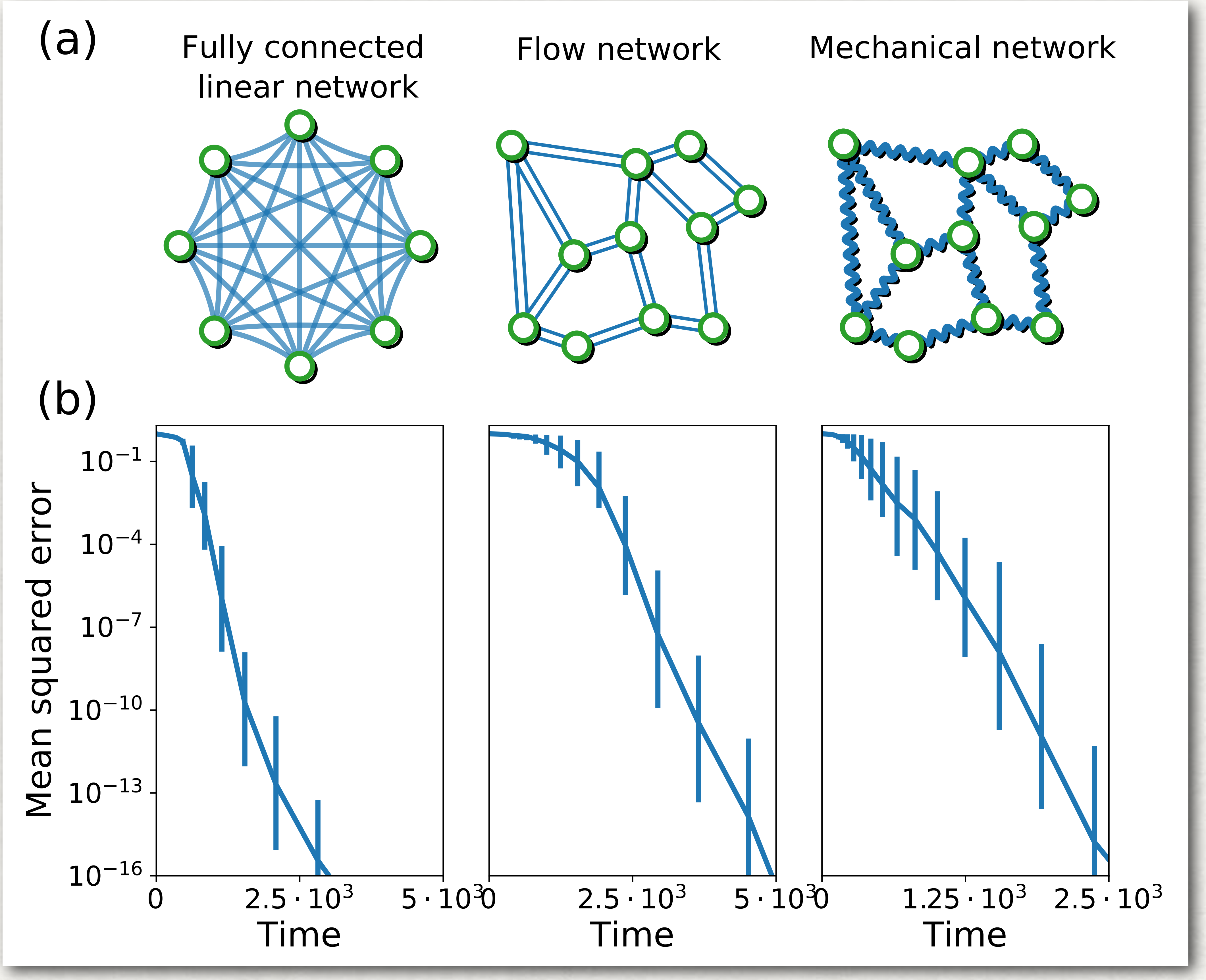
$$\mathcal{F} = \eta^{-1} \left(E^C(\vec{x}^C, \overrightarrow{w}) - E^F(\vec{x}^F, \overrightarrow{w}) \right)$$

Learning Rule: $\delta \overrightarrow{w} = -\alpha \nabla_{\overrightarrow{w}} \mathcal{F}$



This local rule adjusts edges according to their own contribution to the energy.

The learning algorithm works

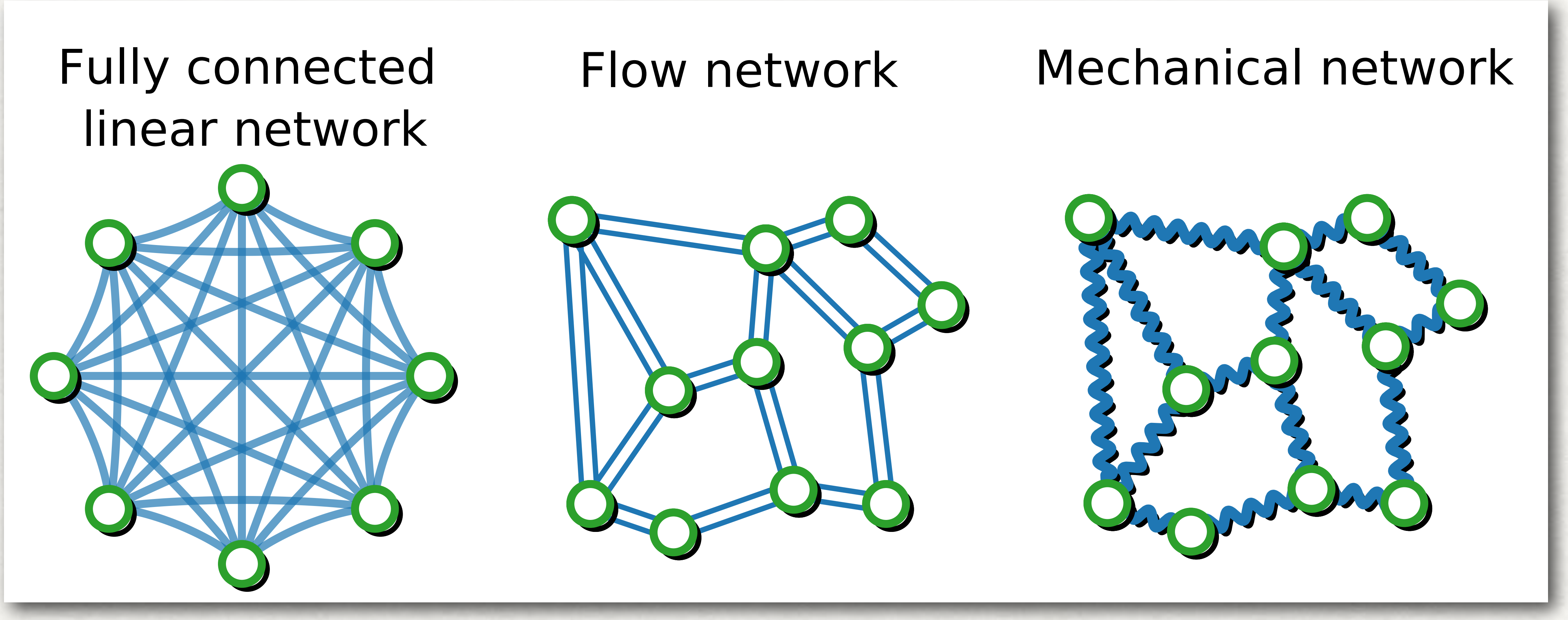


Task: Allostery

1. Start with a random network and choose and input and output node or nodes randomly,
2. Pick an input force of fixed magnitude and random direction
3. Define the allosteric task as requiring a response of a specificed amplitude at the output node
4. Train!

Similar results for other tasks like input classification

The physical effects of training



$$E(\vec{x},\vec{w}) \approx E(\vec{x}^0,\vec{w}) + \frac{1}{2}(\vec{x}-\vec{x}^0)^T H(\vec{w})(\vec{x}-\vec{x}^0)$$

$$H_{ab} = \frac{1}{2}\sum_i \phi_i(w_i)[L_{ai}R_{ib} + R^T_{ai}L^T_{ib}] \equiv \sum_i H^i_{ab}$$

Assume weak input forces and linearize the output constraints: $c = \overrightarrow{A}(\vec{x}^F - \vec{x}^0) - B$

One step of our training protocol then gives the edge weight changes:

$$\delta w_i \approx -\frac{\alpha B \phi'_i}{2}\sum_{ab}(H^{-1}\vec{F})^T_a\,[L_{ai}R_{ib} + R^T_{ai}L^T_{ib}](H^{-1}\vec{A})_b$$

The physical Hessian $H(\overrightarrow{w}, \vec{x}^0(\overrightarrow{w}))$ depends explicitly and implicitly on $\overrightarrow{w}$.

So, defining $\tilde{H}_{ab} = \partial^2 E(\vec{x}, \overrightarrow{w})/\partial x_a \partial x_b$:

$$\delta H = \delta\vec{w}\cdot\left.\frac{d\tilde{H}}{d\vec{w}}\right|_{\vec{x}=\vec{x}^0} = \sum_i \delta w_i \left[\frac{\partial \tilde{H}}{\partial w_i} + \sum_a \frac{\partial \tilde{H}}{\partial x_a}\frac{\partial x^0_a}{\partial w_i}\right]_{\vec{x}=\vec{x}^0}$$

Result

Linearized task: $c = \overrightarrow{A}(\vec{x}^F - \vec{x}^0) - B$ $\quad H_{ab} = \frac{1}{2}\sum_i \phi_i(w_i)[L_{ai}R_{ib} + R^T_{ai}L^T_{ib}] \equiv \sum_i H^i_{ab}$

Use: chain rules, randomness of initial network, weak forces

1. Let $\vec{v}_b$ and λ_b be eigenvectors and eignvalues of the Hessian at any step
2. Let $f_b = \overrightarrow{F} \cdot \vec{v}_b$ and $a_b = \overrightarrow{A} \cdot \vec{v}_b$ be projections of the input force and the (linearized) output constraint vector onto the eigenvectors
3. Use (perturbation theory

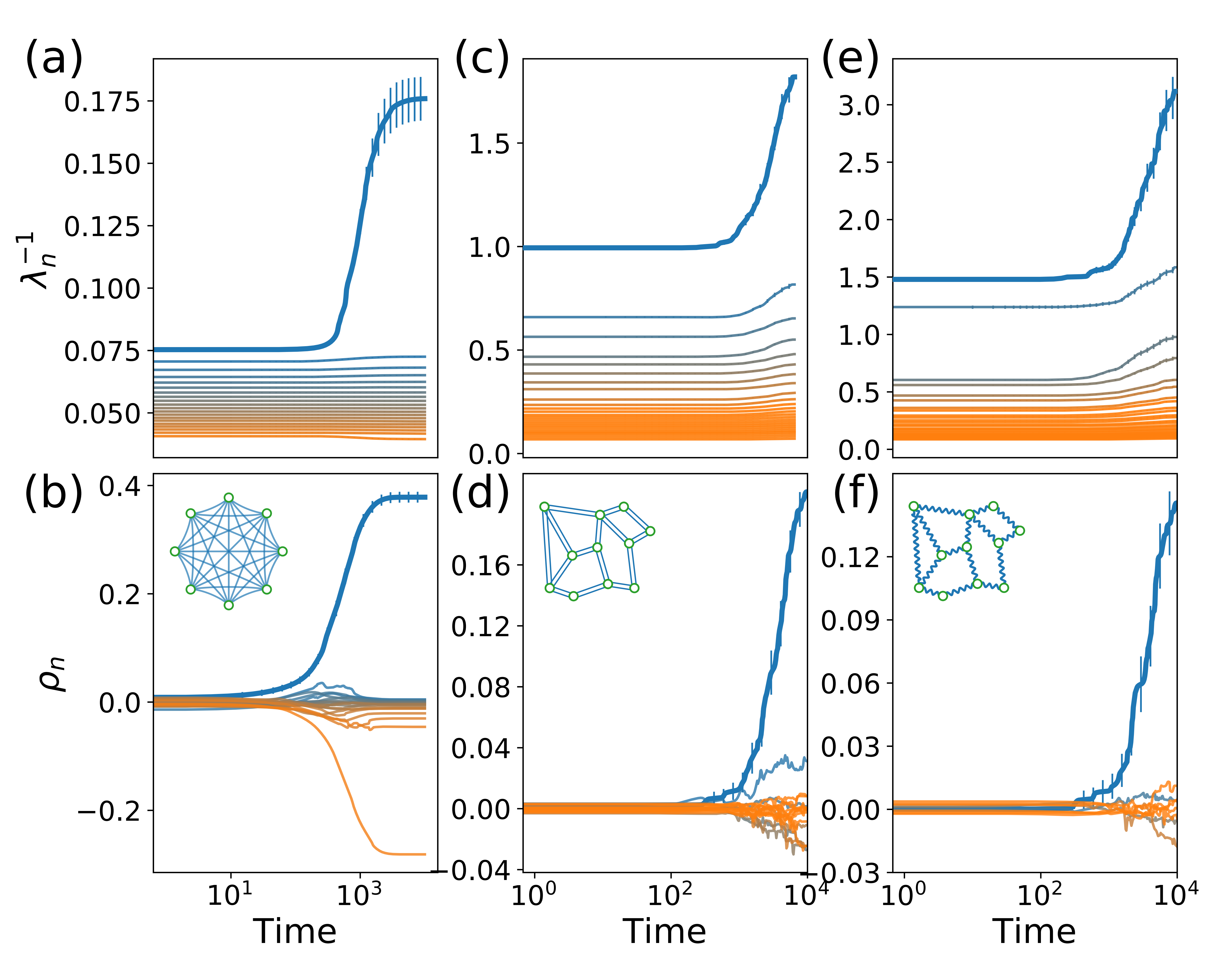
Change in Eigenvalues: $\delta\lambda_n \approx -\alpha B \sum_m Y_{nm} \frac{f_m a_m}{\lambda^2}$

Change in Eigenvectors: $\delta\vec{v}_n \approx \alpha B \sum_{m \neq n} \frac{X_{mn}(f_m a_n + f_n a_m)}{\lambda_m \lambda_n (\lambda_m - \lambda_n)} \vec{v}_m$

X and Y are tensors that depend on the L and R matrices defining the structure of the network

The changes in the eigenspace are larger for smaller eigenvalues, and larger for eigenvalues aligned with the input forces and the output constraints.

The emergence of low eigenmodes aligned with the task



Example: linear networks

$$\delta\lambda_n^{-1} \approx \alpha B \frac{f_n a_n}{\lambda_n^4} = \alpha B \frac{\rho_n}{\lambda_n^4}$$ eigenvalues

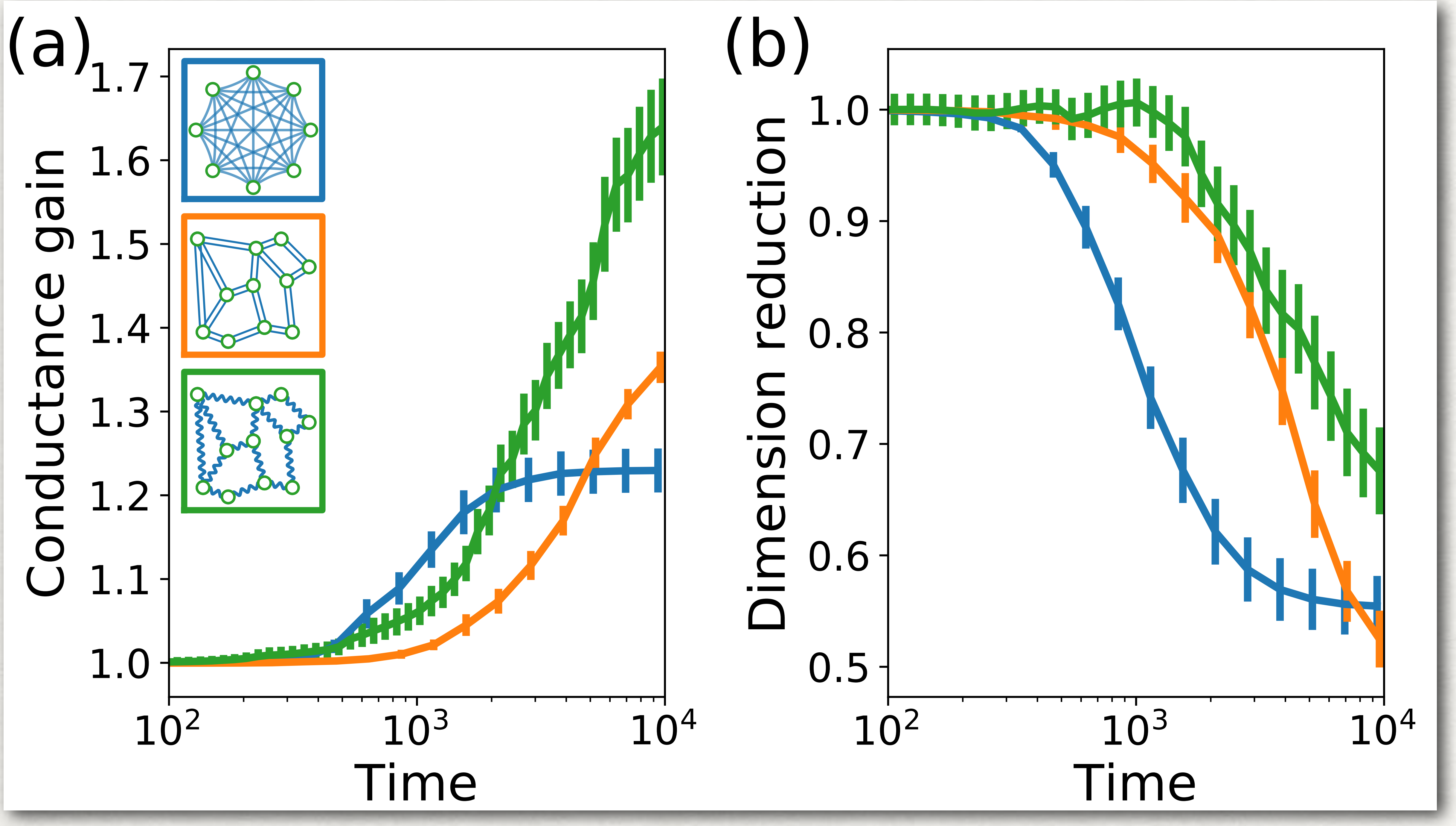
$$\delta\vec{v}_n \approx \alpha B \sum_{m\neq n} \frac{f_m a_n + f_n a_m}{\lambda_m \lambda_n (\lambda_m - \lambda_n)} \delta\vec{v}_m$$ eigenvectors

$$\delta\rho_n \approx \alpha B \sum_{m\neq n} \frac{(f_m a_n + f_n a_m)^2}{\lambda_m \lambda_n (\lambda_m - \lambda_n)}$$ alignment: $\delta(f_n a_n)$

1. One eigenvalue becomes small
2. The associated eigenvector aligns increasingly with both the input force and the task

The network becomes what it learns

The network becomes soft and low dimensional



Test the dimensionality by applying an ensemble of M Gaussian random forces $\{F_m^R\}$ and measuring the response

Effective conductance: $$\bar{g} = M^{-1} \sum_m \frac{|\vec{x}_m^R - \vec{x}^0|}{|\vec{F}_m^R|}$$

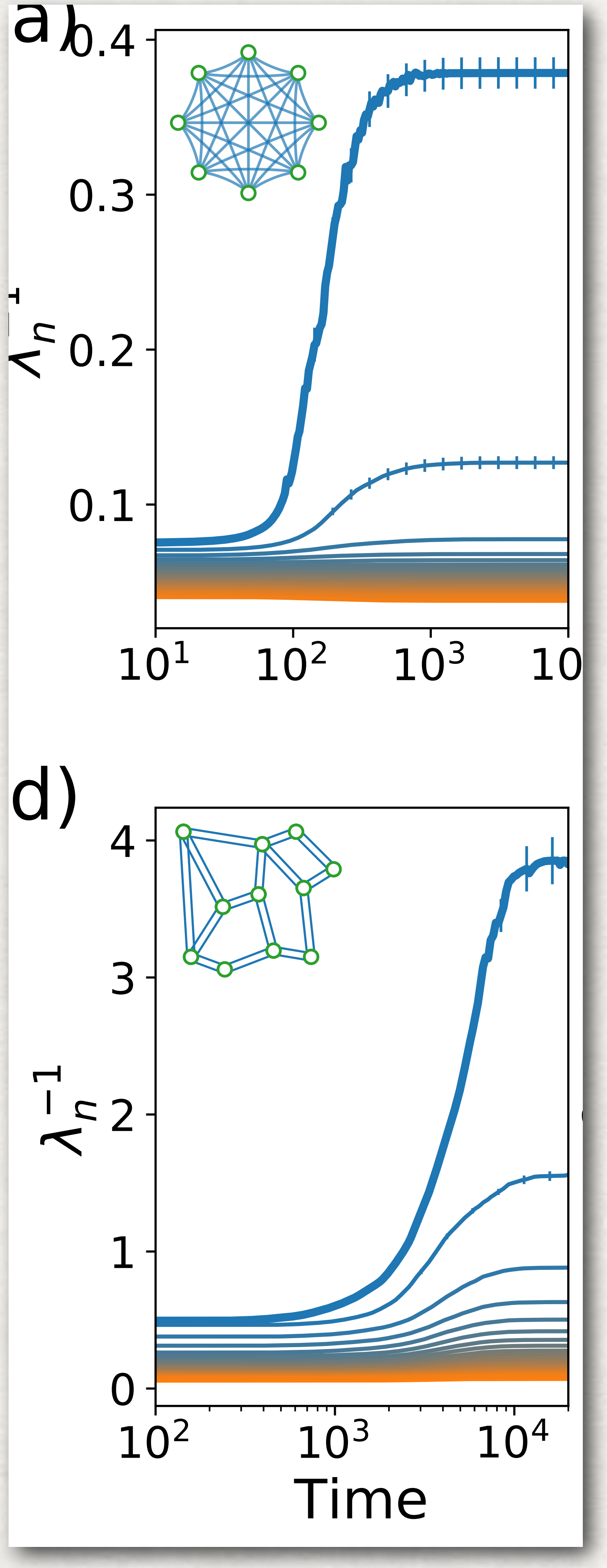
Effective dimension: $$D_{\text{eff}} = \frac{\langle \sum_a p_{am}^2 \rangle^2}{\langle \sum_a p_{am}^4 \rangle}$$

With $p_{am} \equiv \vec{v}_a \cdot (\vec{x}_m^R - \vec{x}^0)$ = projection of responses to random forces onto the eigenvalues

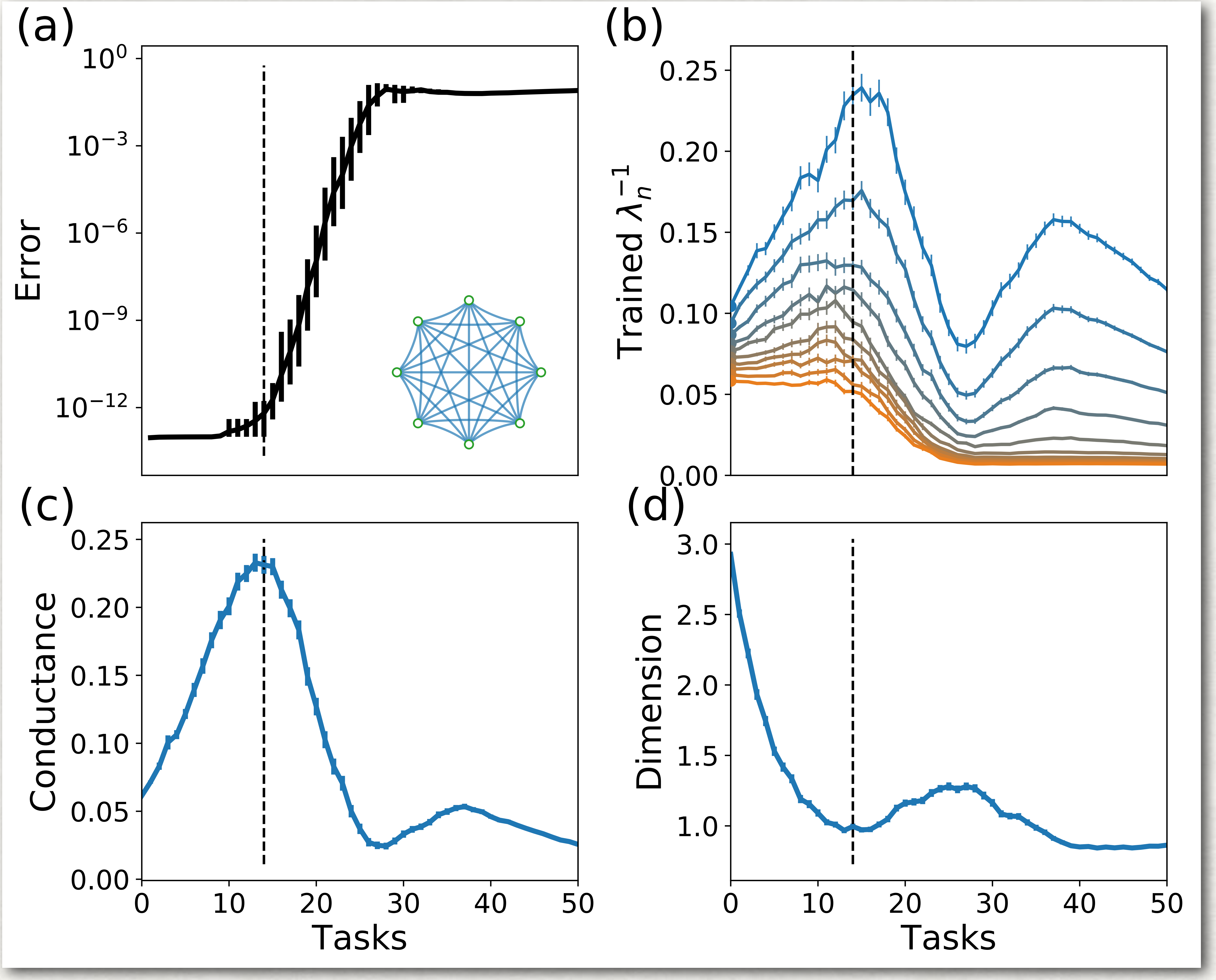
Example: linear networks $$\bar{g} = \sqrt{\sum_a \lambda_a^{-2}} \qquad D_{\text{eff}} = \frac{(\sum_a \lambda_a^{-2})^2}{3 \sum_a \lambda_a^{-4}}$$

Dominated by the lowest eigenvalues

The effects of training multiple tasks



5 tasks



Training beyond capacity

Empirical results

1. Multiple eigenvalues become soft, but not necessarily as many as the number of tasks
2. Beyond the network learning capacity, the error grows, and the network becomes stiff but the effective dimension remains low

Becoming what you learn

1. Network weights reorganizes with learning
2. The response becomes low dimensional
3. The responsive modes become soft and aligned with the input forces and the task
4. Beyond capacity the network remains low dimensional but becomes stiff

Comment 1: Can we discover what a physical network in nature has been trained for by seeing how it responds to random inputs? In fact, this is one of the basic techniques of neuroscience — apply random inputs and analyze circuit responses

Comment 2: Can we construct model networks that contain more features of real neural circuits — heterogeneity of units, asymmetric connections, dynamics, neurogenesis & death