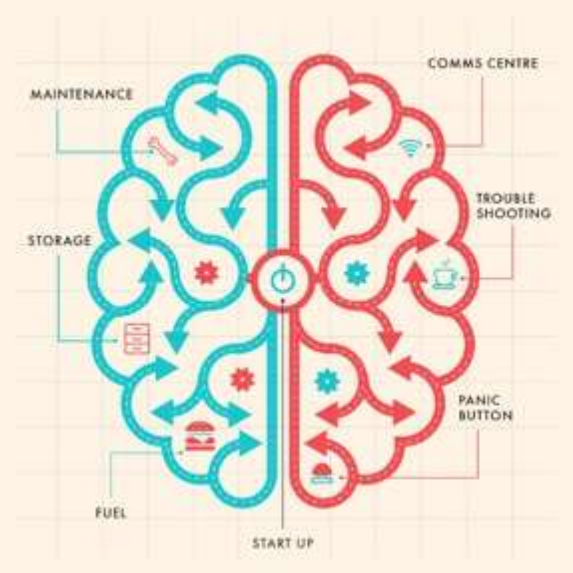
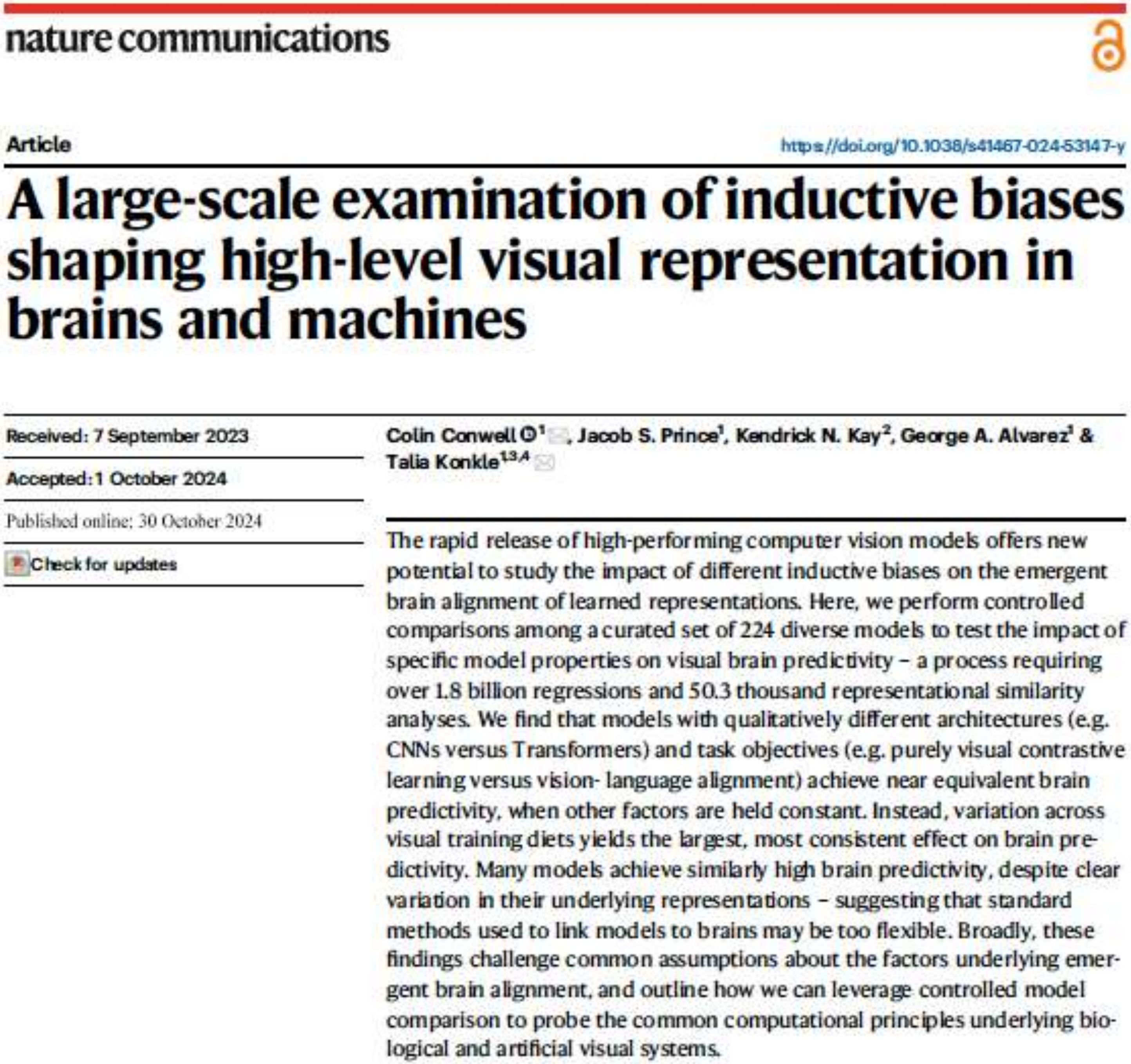
Backpropagation and the Brain

Giuseppe di Pellegrino
Department of Psychology, University of Bologna

g.dipellegrino@unibo.it



Cognition and Neuroscience
Second cycle Degree in Artificial Intelligence – 2024/25



nature communications

Article

https://doi.org/10.1038/s41467-024-53147-y

A large-scale examination of inductive biases shaping high-level visual representation in brains and machines

Received: 7 September 2023

Accepted: 1 October 2024

Published online: 30 October 2024

Check for updates

Colin Conwell[1], Jacob S. Prince[1], Kendrick N. Kay[2], George A. Alvarez[1] & Talia Konkle[1,3,4]

The rapid release of high-performing computer vision models offers new potential to study the impact of different inductive biases on the emergent brain alignment of learned representations. Here, we perform controlled comparisons among a curated set of 224 diverse models to test the impact of specific model properties on visual brain predictivity – a process requiring over 1.8 billion regressions and 50.3 thousand representational similarity analyses. We find that models with qualitatively different architectures (e.g. CNNs versus Transformers) and task objectives (e.g. purely visual contrastive learning versus vision- language alignment) achieve near equivalent brain predictivity, when other factors are held constant. Instead, variation across visual training diets yields the largest, most consistent effect on brain predictivity. Many models achieve similarly high brain predictivity, despite clear variation in their underlying representations – suggesting that standard methods used to link models to brains may be too flexible. Broadly, these findings challenge common assumptions about the factors underlying emergent brain alignment, and outline how we can leverage controlled model comparison to probe the common computational principles underlying biological and artificial visual systems.

Bridging Machine Learning and Neural Circuits

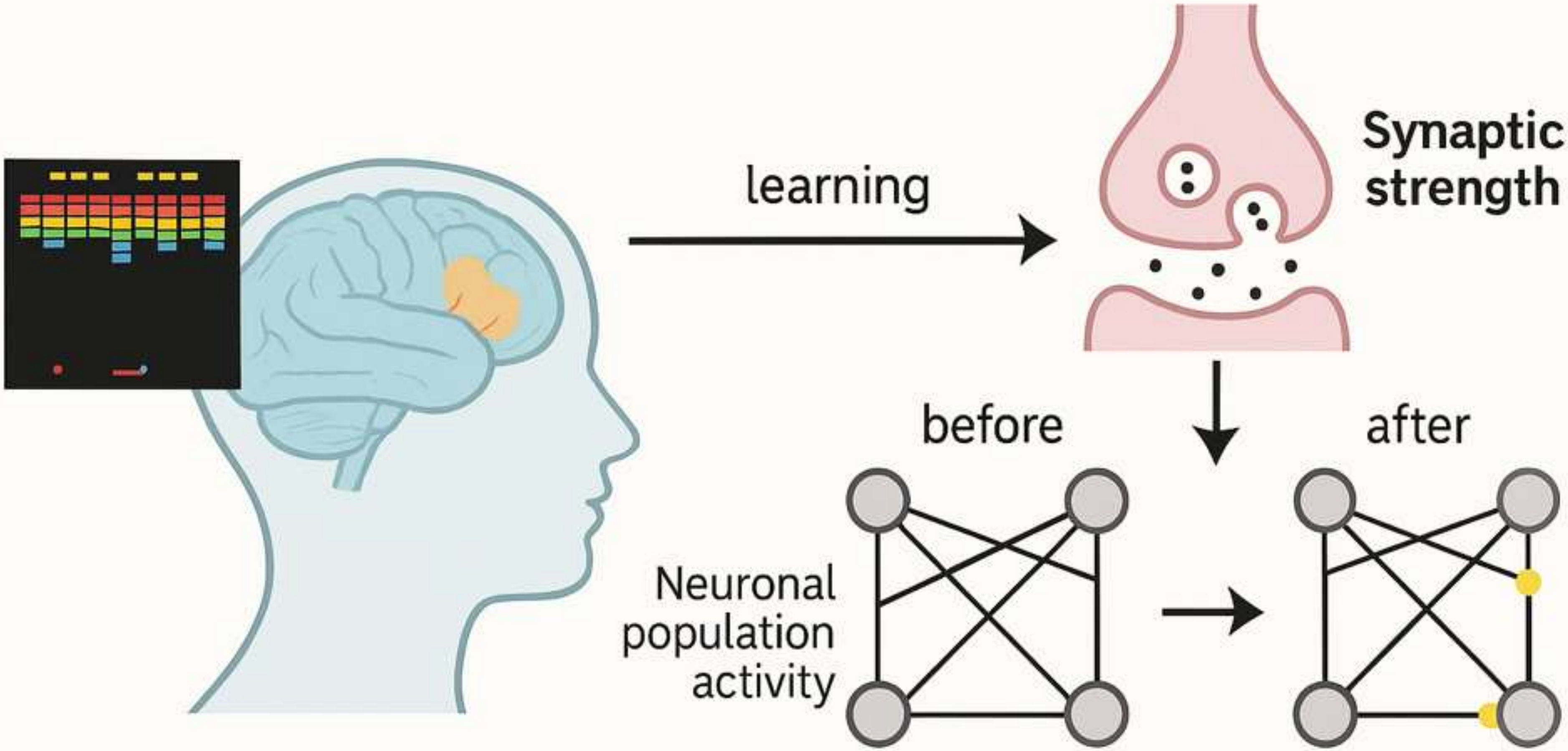
Humans and animals have a remarkable ability to learn adaptive behaviors for thriving in complex environments.

Evolutionary priors (e.g., innate architecture and connectivity) can reduce the amount of learning needed, but many complex human behaviors still require extensive learning (see classic Atari games).

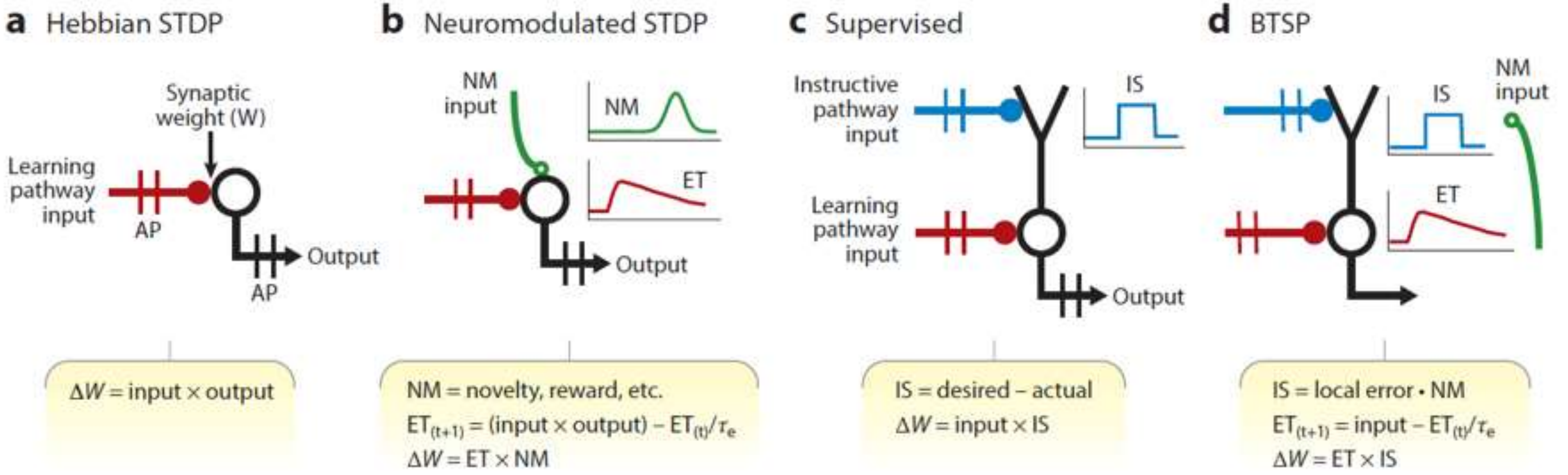
Learning is expected to **modify the input-output functions of neurons** within specific brain regions, thus enabling computations necessary for successful behavior.

Long-term changes in synaptic strength are thought to underlie experience-dependent shifts in neuronal population activity.

Long-term changes in synaptic strength are thought to underlie experience-dependent shifts in neuronal population activity



Synaptic plasticity is nearly ubiquitous throughout brains, and a variety of forms have been reported.


a Hebbian STDP
Synaptic weight (W)
Learning pathway input
AP
Output
AP
ΔW = input × output
b Neuromodulated STDP
NM input
NM
ET
Output
NM = novelty, reward, etc.
$\mathrm{ET}_{(t+1)} = (\text{input} \times \text{output}) - \mathrm{ET}_{(t)}/\tau_e$
$\Delta W = \mathrm{ET} \times \mathrm{NM}$
c Supervised
Instructive pathway input
IS
Learning pathway input
Output
IS = desired – actual
ΔW = input × IS
d BTSP
IS
NM input
ET
IS = local error • NM
$\mathrm{ET}_{(t+1)} = \text{input} - \mathrm{ET}_{(t)}/\tau_e$
$\Delta W = \mathrm{ET} \times \mathrm{IS}$

Abbreviations: BTSP, behavioral timescale synaptic plasticity; ET, eligibility trace; IS, instructive signal; NM, neuromodulator; STDP, spike timing–dependent plasticity, W, synaptic weight change.

Donald Hebb first proposed (1949) that if neuron A repeatedly takes part in firing neuron B, their connection should be strengthened.

Learning occurs via changes in synaptic strength between neurons.

These changes are:

- Causal (based on A leading to B),
- Repetitive (happening repeatedly),
- Gradual (to avoid errors due to noise).

Pioneers of artificial neural networks (Rosenblatt, 1959; Widrow & Hoff, 1960) adapted Hebb’s ideas:

- Weight changes were based on **correlation** between pre- and postsynaptic activity.
- This made learning proportional to the **product of input and output activations**.

$$\Delta w_{i,j} = \beta \cdot f_i(a_i) \cdot f_j(a_j)$$

Discovery of Synaptic Plasticity in Animals

First observed in the hippocampus in the early 1970s (Bliss & Lomo, 1973; Levy & Steward, 1983).
Found that repeated, near-synchronous activation of pre- and postsynaptic neurons leads to:

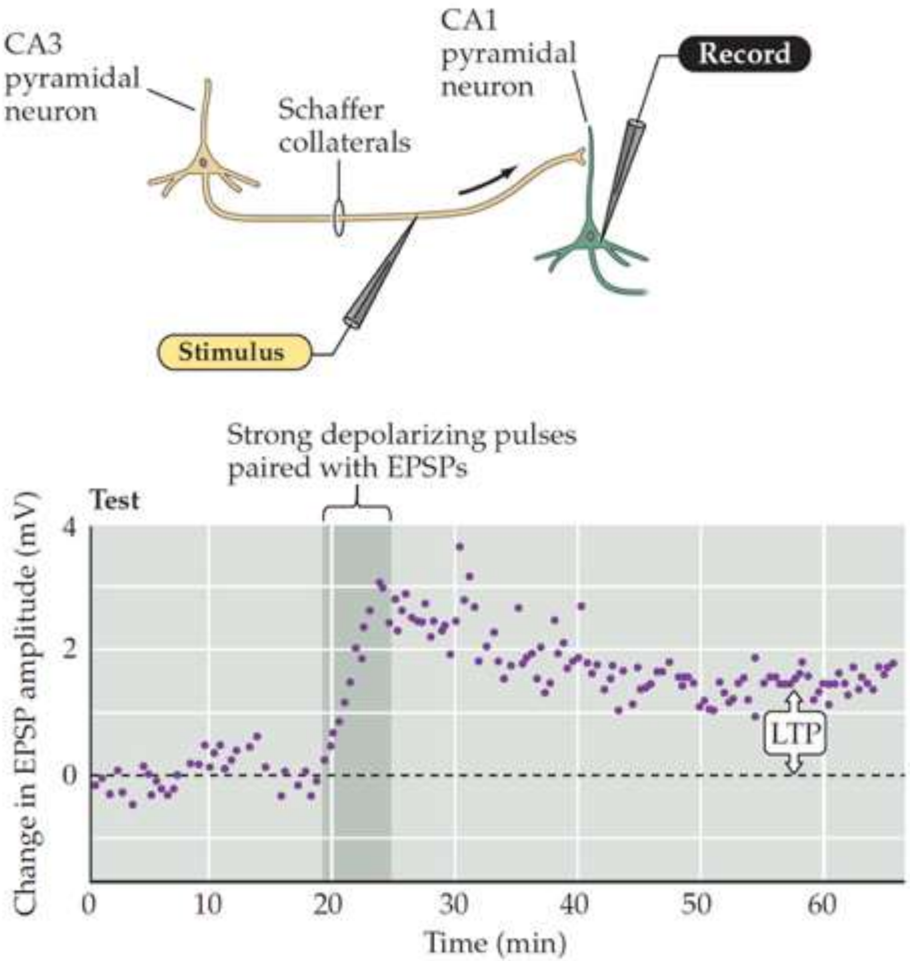
- Increased synaptic strength at only the activated connections.

This phenomenon is called **long-term potentiation (LTP).**

Mechanism: NMDA Receptor:
Central to synaptic plasticity are NMDA-type glutamate receptors:
They require both:

- Glutamate binding (presynaptic activity)
- Membrane depolarization (postsynaptic activity)

This makes them act as **coincidence detectors**.



Pairing presynaptic and postsynaptic activity causes LTP. Single stimuli applied to a CA3 neuron evoke EPSPs in the postsynaptic CA1 neuron. These stimuli alone do not elicit any change in synaptic strength. However, brief polarization of the CA1 neuron's membrane potential (by applying current pulses through the recording electrode), in conjunction with the CA1 stimuli, results in a persistent increase in the EPSPs.

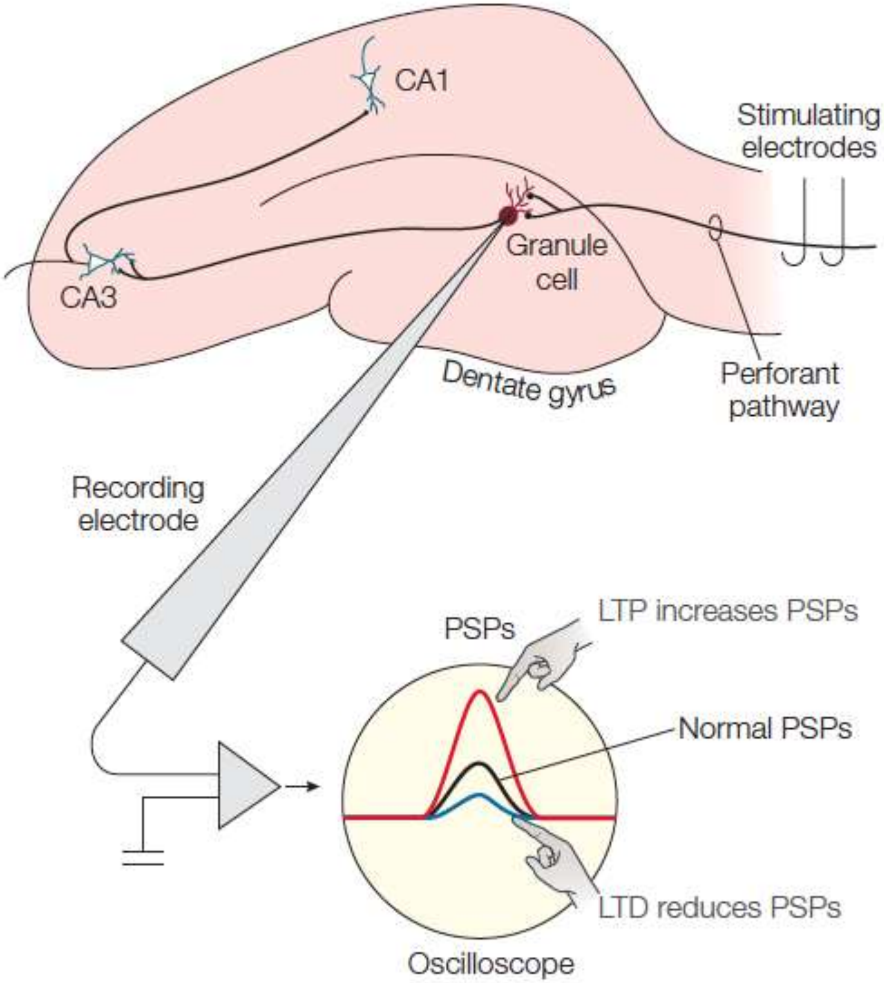
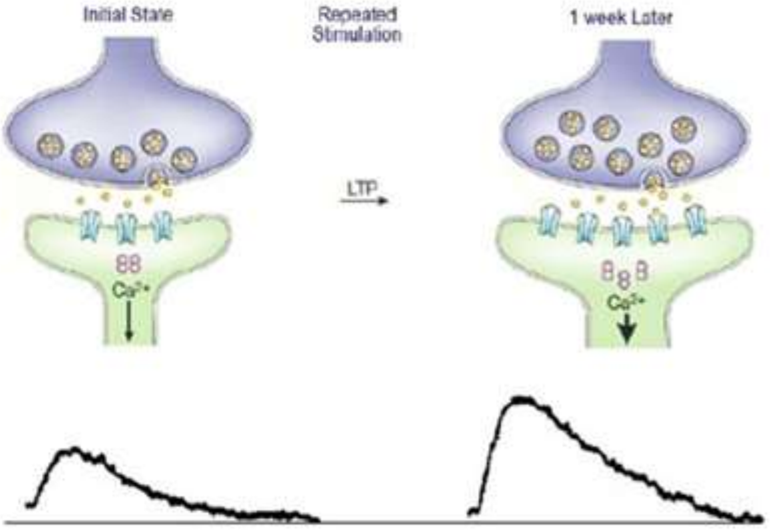
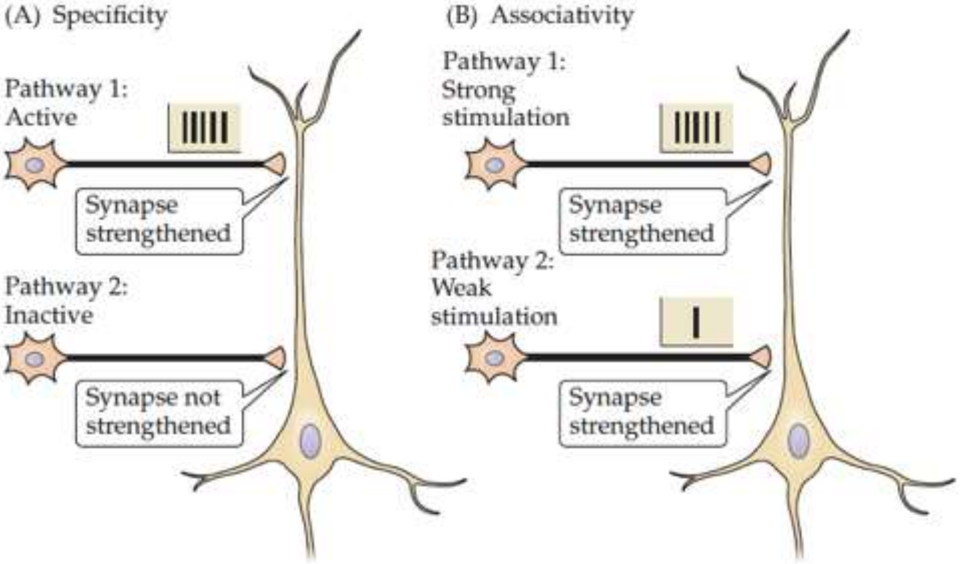


FIGURE 9.38 Stimulus and recording setup for the study of long-term potentiation (LTP) in perforant pathways. The pattern of responses (in millivolts) before and after the induction of LTP is shown as the red curve. The pattern of responses in long-term depression (LTD) is shown as the blue curve. PSPs = postsynaptic potentials.

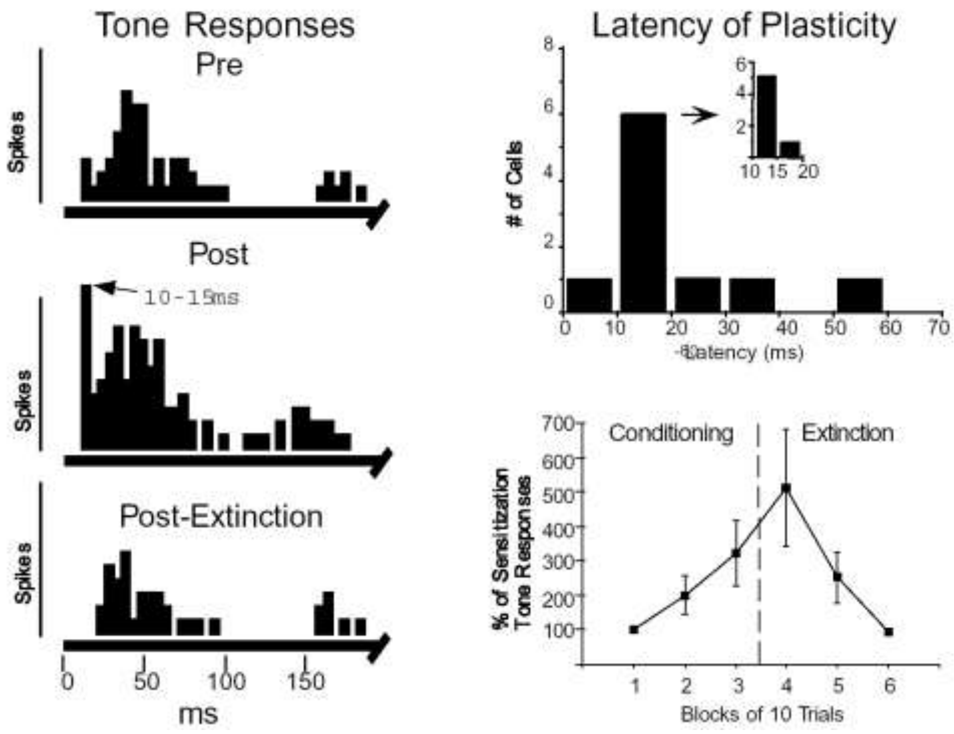


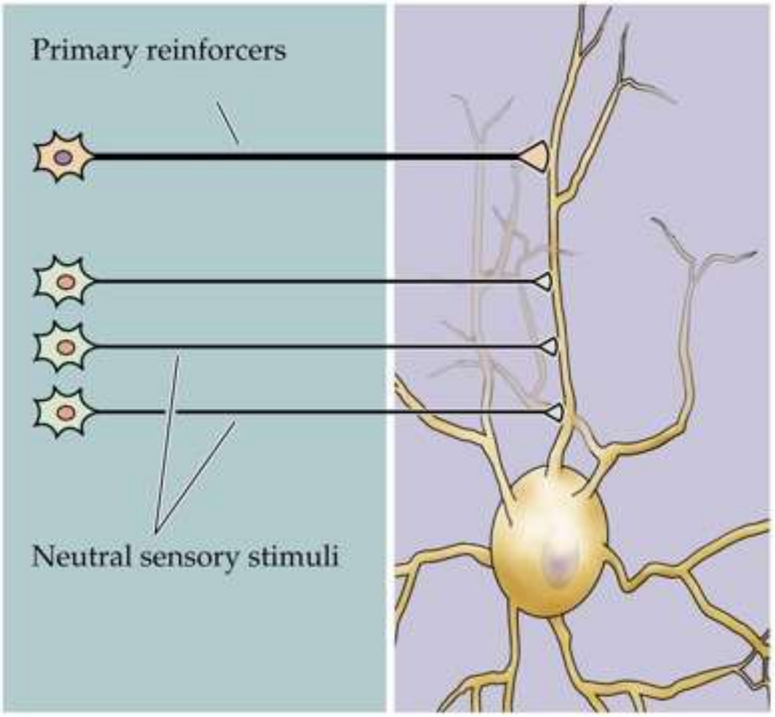


LTP is input specific:
When LTP is induced by activation of one synapse, it does not occur in other, inactive synapses that contact the same neuron

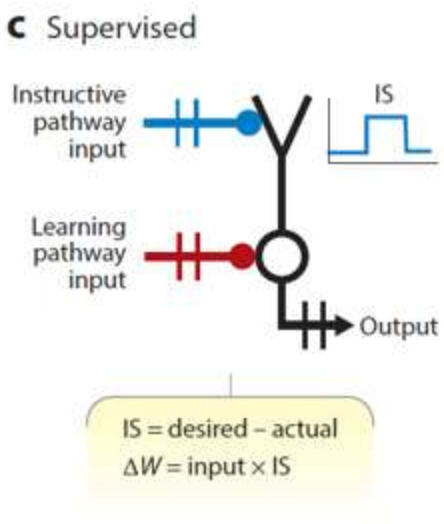
LTP is associative:
Weak stimulation of a pathway will not by itself trigger LTP. However, if one pathway is weakly activated at the same time that a neighboring pathway onto the same cell is strongly activated, both synaptic pathways undergo LTP. Associativity is another consequence of the coincidence detection feature of LTP

Associative LTP in the Amygdala



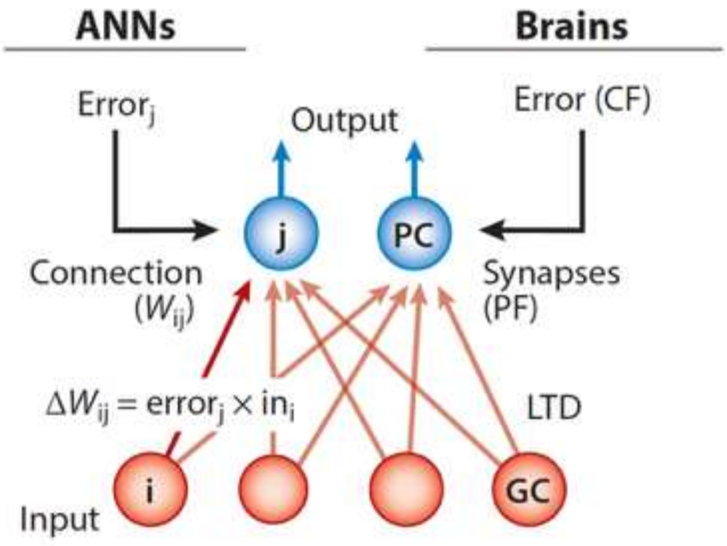


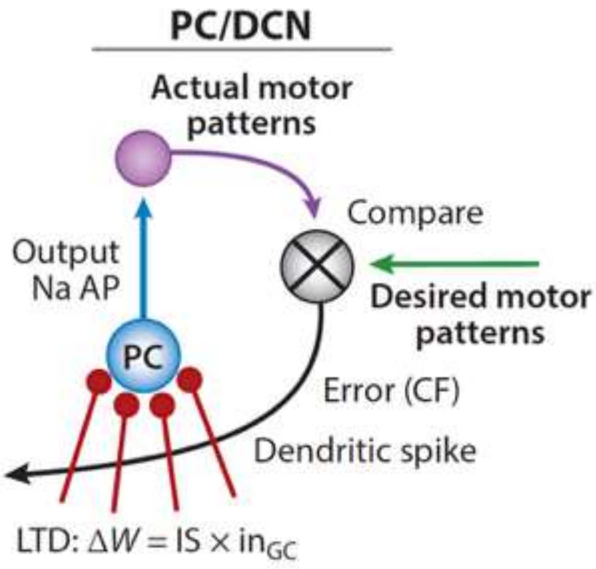
During fear conditioning, cells in the lateral amygdala of rats show plasticity (increased firing rates) during exposure to a conditioned stimulus tone. (Left) Some cells are responsive to tones prior to conditioning (Pre), but their rate of firing increases after conditioning, especially the earliest latency response (10–15 ms after tone onset). This early plasticity goes away after extinction.



The cerebellum works as a **supervised learning** machine involved in learning motor programs or or complex skills

The parallel fibers onto the Purkinje cells (PC) could be viewed as providing sensory prediction while the climbing-fiber (CF) activity provides a representation of sensory prediction errors.
Prediction errors are computed outside the cerebellum by comparing the actual response with the predicted (desired) response and are used as instructive signals to adjust the internal parameters (e.g., synaptic weights) of the system until it learns to generate the desired response.





Synaptic physiology explains how individual synapses undergo modifications.

However, it does not clarify how these changes are coordinated across a network.

Learning is not merely a collection of isolated, miopic synapse-specific events.

Effective learning must account for broader, behaviorally relevant outcomes.

To understand learning in the brain, we must uncover the principles that govern plasticity at the network level.

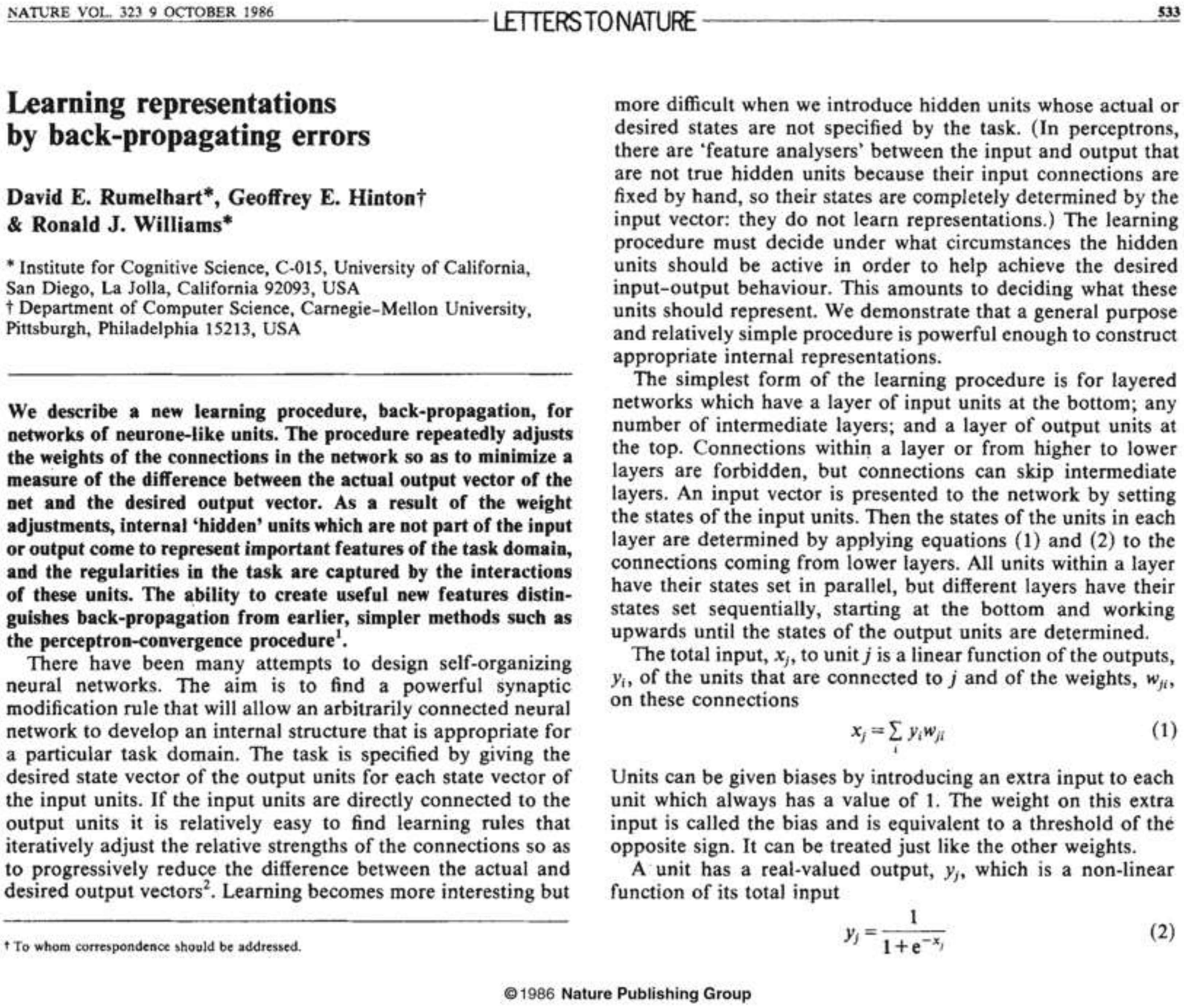
Machine learning explores how to coordinate synaptic updates to optimize artificial neural network performance, free from biological constraints.

Researchers begin by defining the neural network's architecture—number of neurons and connectivity—often using deep networks due to their effectiveness.

An error function is defined to measure how far the network is from its goals.

Learning algorithms that compute synaptic changes that reduce
the error.

Backpropagation of error (Rumelhart et al., Nature, 1986) is the algorithm most often used to train deep neural networks and is the most successful learning procedure for these networks.



Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].

There have been many attempts to design self-organizing neural networks. The aim is to find a powerful synaptic modification rule that will allow an arbitrarily connected neural network to develop an internal structure that is appropriate for a particular task domain. The task is specified by giving the desired state vector of the output units for each state vector of the input units. If the input units are directly connected to the output units it is relatively easy to find learning rules that iteratively adjust the relative strengths of the connections so as to progressively reduce the difference between the actual and desired output vectors[2]. Learning becomes more interesting but more difficult when we introduce hidden units whose actual or desired states are not specified by the task. (In perceptrons, there are 'feature analysers' between the input and output that are not true hidden units because their input connections are fixed by hand, so their states are completely determined by the input vector: they do not learn representations.) The learning procedure must decide under what circumstances the hidden units should be active in order to help achieve the desired input–output behaviour. This amounts to deciding what these units should represent. We demonstrate that a general purpose and relatively simple procedure is powerful enough to construct appropriate internal representations.

The simplest form of the learning procedure is for layered networks which have a layer of input units at the bottom; any number of intermediate layers; and a layer of output units at the top. Connections within a layer or from higher to lower layers are forbidden, but connections can skip intermediate layers. An input vector is presented to the network by setting the states of the input units. Then the states of the units in each layer are determined by applying equations (1) and (2) to the connections coming from lower layers. All units within a layer have their states set in parallel, but different layers have their states set sequentially, starting at the bottom and working upwards until the states of the output units are determined.

The total input, x_j, to unit j is a linear function of the outputs, y_i, of the units that are connected to j and of the weights, w_{ji}, on these connections

$$x_j = \sum_i y_i w_{ji} \qquad (1)$$

Units can be given biases by introducing an extra input to each unit which always has a value of 1. The weight on this extra input is called the bias and is equivalent to a threshold of the opposite sign. It can be treated just like the other weights.

A unit has a real-valued output, y_j, which is a non-linear function of its total input

$$y_j = \frac{1}{1+e^{-x_j}} \qquad (2)$$

† To whom correspondence should be addressed.

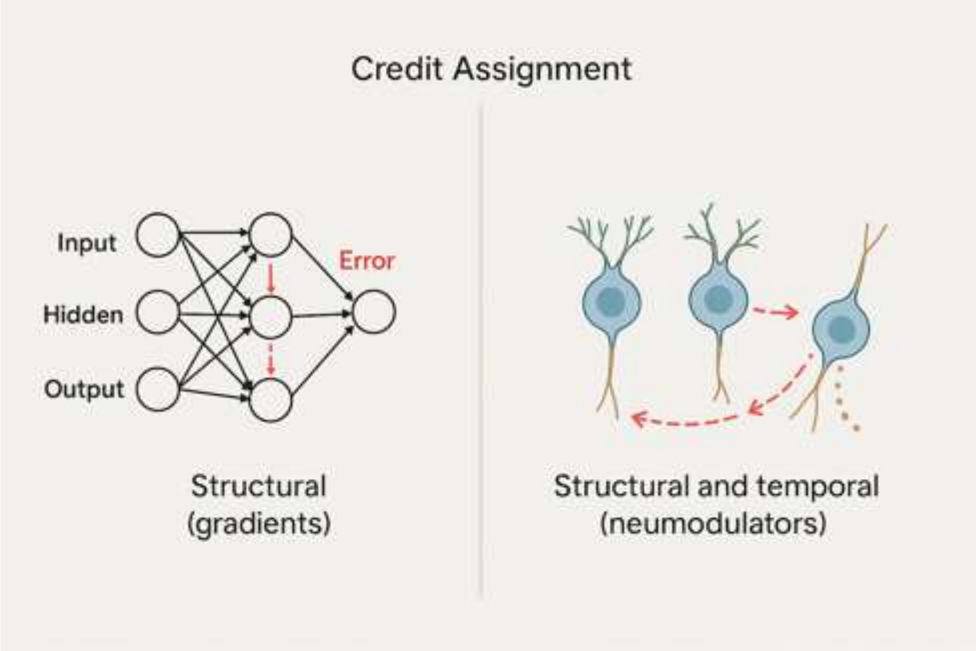
Credit Assignment in Learning Algorithms

A wide variety of **learning algorithms** have been developed for both the **brain** and **artificial neural networks (ANNs)**.

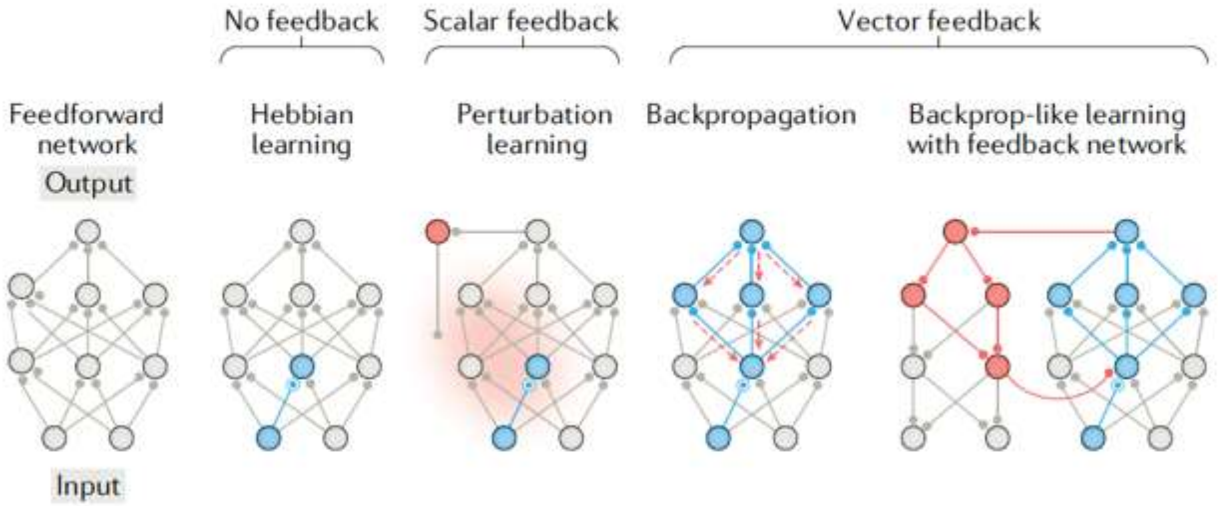
How do they assign credit across multiple layers of neurons?

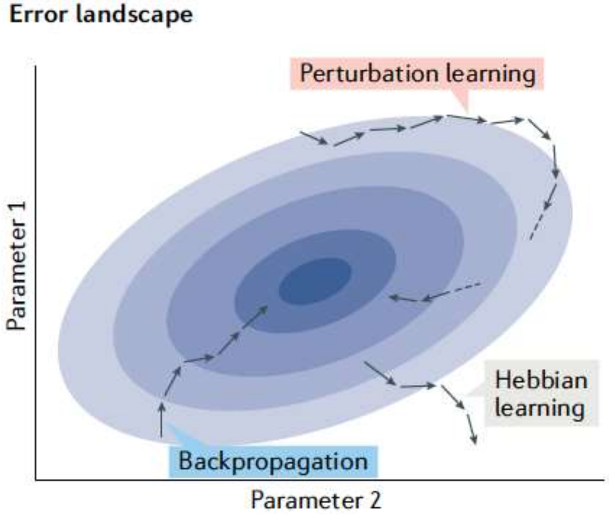
This is crucial for explaining:

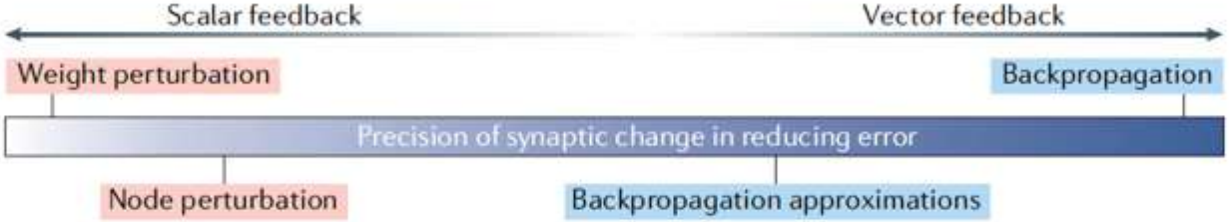
- How layered neural systems (like the cortex or deep networks) **learn from feedback**.
- How changes in early-layer weights contribute to outcomes several steps downstream.



Spectrum of learning algorithm







Without feedback, synaptic parameters wander randomly on the error surface.

Scalar feedback does not require detailed feedback circuits, but it learns slowly.

Precise vector feedback via backprop learns quickly.

Backpropagation of error is the algorithm most often used to train deep neural networks and is the most successful learning procedure for these networks.

Perturbation learning

Snaptic weights are adjusted based on how random changes affect the global performance

Briefly, compute baseline error: E, add small Gaussian noise δ to a weight Wij, recompute error
Finally, one accepts the modified synaptic weight if the network's new error is less than the baseline error.
This procedure can be implemented simply by a learning rule that broadcasts a global scalar (i.e., total error change) representing the overall change in performance of the network

- **Simple but inefficient** — slow to scale to deep networks
- Inspired by possible **biological mechanisms** (e.g., "hedonistic synapses" responding to gobal reward signals, such as dopamine, Seung Neuron, 2021)

No known method based solely on perturbation (without gradient computation) has been able to train a deep network to high performance on tasks like ImageNet classification.

Backpropagation: Efficient Credit Assignment in Neural Networks

Solves inefficiency of perturbation methods by **computing**, **not measuring**, how weight changes affect error.
Leverages exact causal relationships between weights and outputs to calculate precise gradients.

Uses the chain rule of calculus to efficiently compute how each weight affects the network's total error.
Computes all weight updates simultaneously — with similar cost to one forward pass.
Implements error propagation recursively, layer by layer — leading to the name "backpropagation."

Once error signals are assigned to neurons, each neuron adjusts its incoming weights to reduce error.

Works across learning paradigms:

- Supervised learning (target vs. prediction)
- Reinforcement learning (e.g., temporal difference errors)
- Unsupervised learning (e.g., prediction error)

Does not require external labels — error signals can be internally generated.

Two Key Features of Backpropagation (Biologically Relevant)

Synapse-Specific Learning

Backprop prescribes specific changes to each synapse, similar to biological mechanisms like spike-timing-dependent plasticity (STDP).

Feedback for Error Signals

Requires feedback pathways to transmit error signals to deep neurons — analogous to feedback loops observed in the brain.

Feedback pathways are widespread in the cortex and modulate the activity of feedforward neurons.

- Top-down cortico-cortical connections (e.g., V2 → V1 in vision)
- Cortico-thalamo-cortical loops that integrate higher-order signals

Thus, if feedback modulates spiking, and spiking drives synaptic plasticity,
then feedback must influence learning

Note however that feedback are possibly **coarse, delayed, or scalar**, not detailed and layer-specific as prescribed by backpropation.

Does the Brain Use Backpropagation?

No direct evidence that the brain uses backprop-like algorithms.

However, **models trained with backprop** can replicate neural responses in areas like:

- Posterior parietal cortex
- Primary motor cortex
- Recent visual cortex models continue this trend.

These models are **not perfect** — they miss some aspects of **human object recognition**.

Still, their success suggests that **backprop-like learning cannot be ruled out** on representational grounds (*Cadieu et al.*).

Emerging evidence shows:

- **Cortical neurons (layers 2/3)** may compute **prediction errors**.
- **Neural dynamics in vision** are consistent with **hierarchical error processing**.

Problems with backprop

Backprop’s computations:

$$\Delta W_l = -\eta \frac{\partial E}{\partial W_l} = -\eta \boldsymbol{\delta}_l \mathbf{h}_{l-1}^{\top}$$

Weight Update

where

$$\boldsymbol{\delta}_l = \mathbf{e}_l \circ \mathbf{f}'(\boldsymbol{a}_l) = (W_{l+1}^{\top} \boldsymbol{\delta}_{l+1}) \circ \mathbf{f}'(\mathbf{a}_l)$$

Error Signal (δ) Backpropagation

It is perhaps worth noting that, when presented in the form:

$$\Delta W_l = -\eta(\mathbf{e}_l \circ \mathbf{f}'(\boldsymbol{a}_l))\mathbf{h}_{l-1}^{\top}$$

the update can be seen as a local Hebbian-like rule —where the postsynaptic activity is replaced by $f'(a_l)$ — that is modulated by a third factor, e_l, which is computed via feedback connections.

1. There is no obvious source for the supervision signal

The brain likely does not need a perfect “supervision signal” like in supervised learning.

Backpropagated signal does not need to be a difference between
an output and a supervised target.
The signal can also be a temporal difference error in reinforcement learning,
or a reconstruction or prediction error for an unsupervised algorithm.

Instead of using a single supervision signal, the brain may rely on a **combination of weak, distributed, and local signals** that, together, guide powerful learning.

2. Unrealistic Models of Neurons in ANNs

Continuous vs. Spiking Signals
Artificial neurons output continuous values (e.g., between 0 and 1), typically interpreted as firing rates.

Biological neurons, however, communicate using discrete spikes (action potentials) — not continuous values.

Backpropagation Requires Differentiability
The core of backpropagation involves computing derivatives (gradients).

But for spiking neurons, there's no smooth output function to differentiate — a spike is an all-or-none event.

This makes it non-trivial (and in many cases impossible) to apply backprop directly to spiking neural networks (SNNs).

3. Challenges of Signed Error Signals in the Brain

Backpropagation relies on **signed error signals** (δ) that can vary wildly in magnitude — leading to **exploding or vanishing gradients**.

While some evidence for signed error signals exists (e.g., in the **cerebellum**), it's **unclear how such signals could be propagated in deep cortical networks**.

One idea: use separate **"error neurons"** where firing rate encodes positive or negative errors — but this requires complex **coordination** across layers and feedback paths.

The brain may instead use alternative mechanisms that **avoid explicit backpropagation of error signals**.

4. Backprop demands synaptic symmetry in the forward and backward paths

In backpropagation, each feedback connection must match the strength of its corresponding feedforward connection (i.e., symmetric weights: $W^\top$).

One early biological proposal was an "error delivery network" where each backward neuron matched a forward neuron to transmit update signals.

Another idea was that errors could travel backward along the same feedforward axons (retrograde signaling) with equal strength — but this was rejected because retrograde signaling is too slow.

The core difficulty is known as the “**weight transport problem**”

Backprop requires the same weight value in two different pathways, which is biologically implausible.

Some past work suggested using symmetric learning rules to enforce matching weights, but the cortex lacks the required symmetric, point-to-point connections to make this work.

Feedback Alignment

Backpropagation’s symmetry requirement (matching feedforward and feedback weights) is biologically implausible.

However, recent work shows that symmetry requirement (matching feedforward and feedback weights) is not necessary for learning.

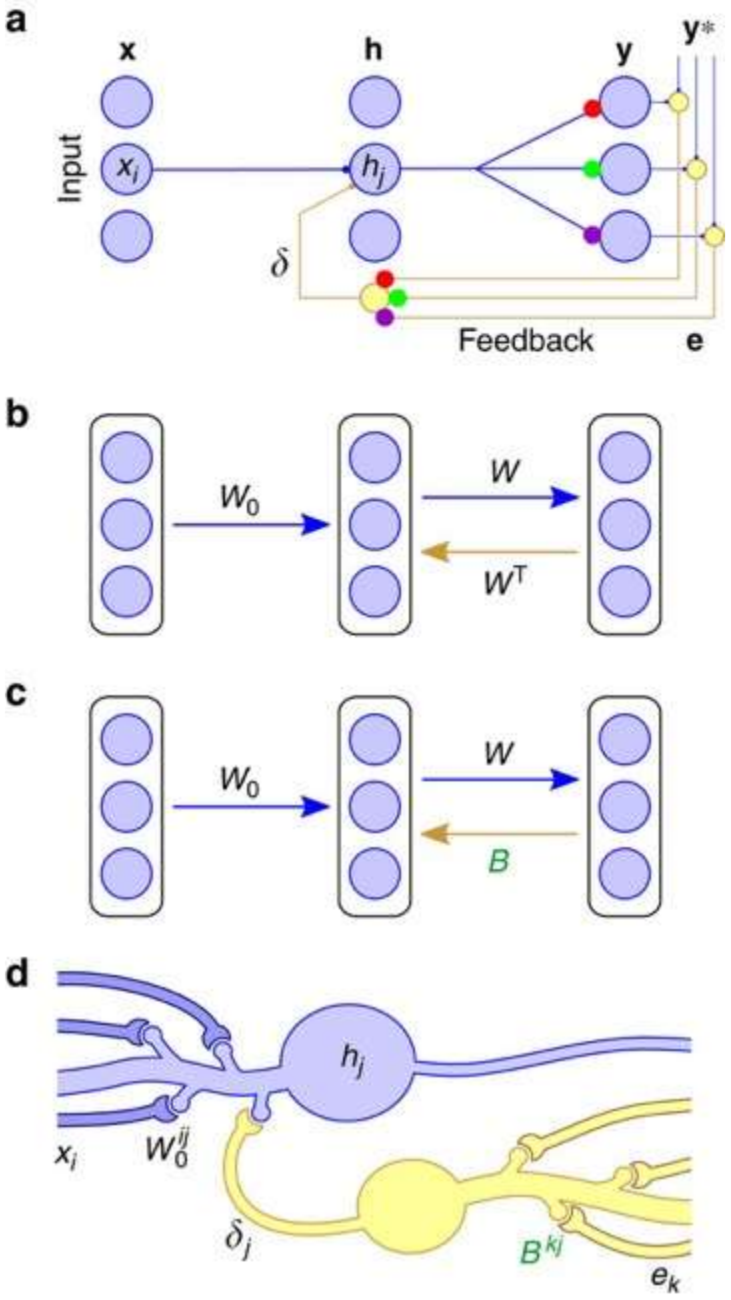
Networks can learn using fixed, random feedback weights instead of symmetric ones.

Over time, the feedforward weights align with the fixed feedback weights.

This leads to "fake" error signals (from random feedback) that still guide the network to reduce the true error.

Despite being incorrect, **these fake signals nudge the network in the right direction**.

While random feedback may not work in very deep networks, later studies show that simple local learning mechanisms can adapt the feedback pathway, enabling effective training even on complex, deep tasks.



(a) The backprop learning algorithm requires that neurons know each others' synaptic weights, for example, the three coloured synapses on the feedback cell at the bottom must have weights equal to those of the corresponding coloured synapses in the forward path.

(b) Backprop computes teaching, or modulator, vectors by multiplying the error vector **e** by the transpose of the forward weight matrix W, that is, **δ=W^{T}e**.

(c) Feedback alignment method replaces W^T with a matrix of fixed random weights, B, so that **δ=B e**. Thus, each neuron in the hidden layer receives a random projection of the error vector.

(d) The brain might use **top-down modulatory signals** that are **statistically helpful (loss reduction)**, even if not mathematically perfect

Lillicrap et al., Nat. Comm., 2016

In backpropagation:
The error signal at layer *l* computed using the **transpose of the next layer's weights**:

$$\delta_l = (W_{l+1}^\top \delta_{l+1}) \circ f'(a_l)$$

This means the feedback path must use **exact copies** (or transposes) of feedforward weights.

In the brain, this would require neurons to know the exact strengths of connections they don't even receive — known as the **weight transport problem**.

In 2016, Lillicrap et al. showed t $\delta_l \approx (B_{l+1}\delta_{l+1}) \circ f'(a_l)$

a **fixed, random matrix,** works surprisingly well.

Although the random feedback does not match the forward weights at all, the network still learns — and in many cases, performs comparably to backprop.

Over time, the feedforward weights naturally align to the direction of the random feedback signals.

This means the "fake" gradients given by the random feedback end up pointing roughly in the same direction as the true gradients.

This emergent alignment between the directions of updates is what gives the method its name: feedback alignment. FA works well in shallow or moderately deep networks. In very deep or complex networks, random feedback alone often fails.

5. Lack of Local Error Representation

How the Brain Learns (Locally)

•Biological synapses (connections between neurons) adjust their strength using **local information only**, such as:

- The firing activity of the **pre- and postsynaptic neurons**.
- The timing between their spikes.
- Possibly local chemical signals (e.g., calcium, dopamine, neuromodulators).

This is known as **local learning:** Biological synapses do not have access to what is happening in **distant downstream neurons**.

How Backpropagation Works (Non-Locally)

•In backpropagation:

- The update to a synaptic weight (say, between neuron A and B) depends not just on A and B, but on **all the neurons downstream of B**.
- The error signal is computed by **propagating derivatives backward** through multiple layers — a **global computation**.

Each weight update in backprop depends on a **non-local error** that's **not directly available** at the synapse.

Why This Is a Problem (Biologically)

- In the brain, there is no clear mechanism for sending this kind of global error signal back to individual synapses with precise, layer-by-layer information.
- Backprop assumes that every synapse "knows" how a small change in its strength would affect the overall output error.
- But real synapses can only access local activity, not this global credit assignment.

This mismatch is a major historical criticism of backprop as a biological learning model.

A local error representation would mean that each synapse can compute how wrong its output is using only local, available signals.
This is what biological learning likely requires — and what backprop fails to offer directly.

Many researchers have focused on replacing global error signals with local approximations.
Examples include:

Hebbian learning (correlation-based)
Spike-timing–dependent plasticity (STDP)
Feedback alignment
Predictive coding
Dendritic error compartmentalization (where different parts of a neuron represent different signals)

6. Feedback in brains can drive neural activity, not just modulate it

In ANNs, feedback connections deliver error signals (δ).
These error signals do not influence neuron activity during inference.
Their only role is to guide synaptic weight updates during learning.

In other words: feedback is silent during inference, active only during training.

In the Brain, feedback connections play an active, real-time role in shaping neural activity.
They directly modulate the responses of neurons during perception and cognition.

- Top-down control: enhancing or suppressing neural responses (e.g., in attention).
- Contextual modulation: adjusting perception based on expectations or goals.

Feedback in ANNs is used solely for learning, while feedback in the brain is used continuously for computation — not just for training.

The NGRAD (Neural Gradient Representation by Activity Differences) hypothesis

Despite key differences, the brain might still implement core principles of backpropagation.
This can happen without replicating its exact algorithmic structure (e.g., symmetric weights or external error variables).

- Feedback connections in the brain might induce changes in neural activity.
- The difference between feedforward and feedback-driven activity could represent an error signal.
- These local activity differences could be used to update synapses — mimicking backprop-like gradient descent.

NGRAD sidesteps major biological issues with backprop, such as:
The weight transport problem
Lack of local error signals
Absence of differentiable spike outputs

NGRAD suggests that the core computational goal of backprop — credit assignment — might be approximated in the brain through local activity differences, rather than exact gradient calculations.

Core Shift in Approach

Unlike standard backprop, which propagates explicit error derivatives, NGRAD
Relies on differences between two activity states to drive learning.

These differences can be temporal (before and after feedback) or state-based (e.g., feedforward vs. feedback-influenced).

Two-Phase Inference

Learning is based on comparing neural activity during:

1. A feedforward pass (initial response to input), and
2. A feedback-modulated pass (adjusted by target or context).

The difference between these phases encodes an implicit error signal.

Synapses use this local change in activity to compute weight updates, in a way that approximates backpropagation.

Advantages of the NGRAD Approach

No need for two types of information (activations and derivatives):

All learning relies on local, biologically plausible signals.

Higher-level activity (context, target, modality) nudges lower-level neurons toward more consistent outputs.

This induced shift in activity acts like a teaching signal, guiding learning at earlier layers.

NGRAD aims at explaining how the brain might approximate backpropagation without relying on biologically implausible mechanisms like symmetric weights or non-local error signals

Use a Multilayer Autoencoder to explore the NGRAD Hypothesis

The autoencoder structure:

- Has both feedforward (encoder) and feedback (decoder) pathways.
- Can be trained to reconstruct its input using layer-by-layer mappings.
- Is a natural way to explore how feedback signals can influence intermediate representations.

This architecture mimics the recurrent, layered structure of the cortex, making it a biologically motivated testing ground for NGRAD.

How NGRAD Works in the Autoencoder

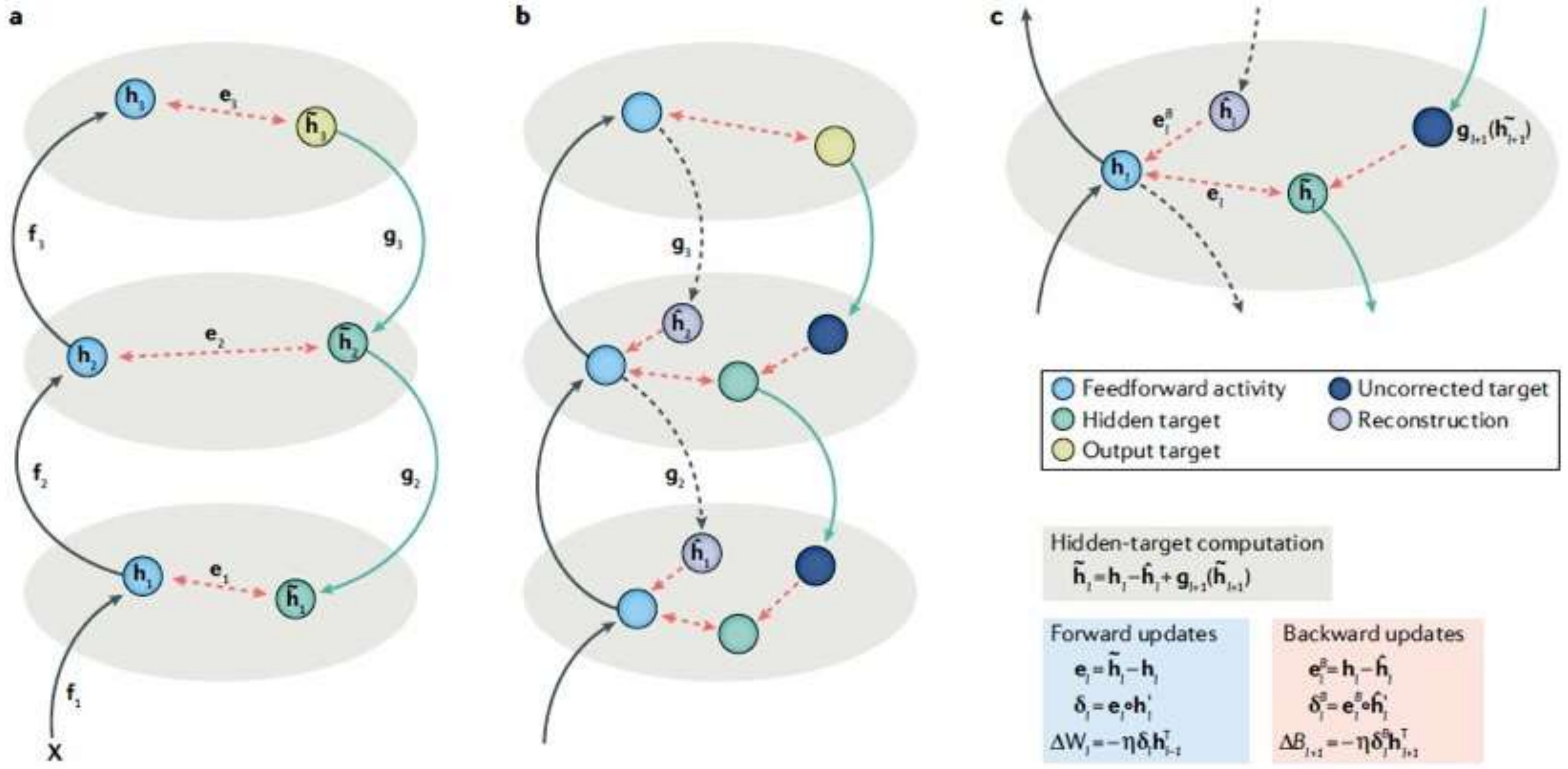
1. Feedforward phase: Input flows through encoder layers to generate hidden representations h1, h2, h3,…
2. Feedback phase: The decoder pathway uses inver**se feedback functions** gl to reconstruct lower-layer activities. These reconstructions serve as targets:
3. Activity difference as error: Each neuron compares its current activity to the target activity:

This difference is used to drive local synaptic updates, much like gradients in backprop.

4. Difference Target Propagation (DTP): Since reconstructions (inverse functions) are not perfect, DTP corrects for the inaccuracy of the feedback by adjusting the targets

Overall

- It shows that **local activity differences** (not global error derivatives) can guide learning.
- Feedback does not need to be symmetric — just **influential** enough to nudge activity in useful directions.
- This process can approximate the function of backprop **using only local, biologically plausible signals**.


a
h_3
e_3
$\tilde{h}_3$
f_3
g_3
h_2
e_2
$\tilde{h}_2$
f_2
g_2
h_1
e_1
$\tilde{h}_1$
f_1
x
b
g_3
$\hat{h}_2$
g_2
$\hat{h}_1$
c
e^B_l
$\hat{h}_l$
$g_{l+1}(\tilde{h}_{l+1})$
h_l
e_l
$\tilde{h}_l$
Feedforward activity
Hidden target
Output target
Uncorrected target
Reconstruction
Hidden-target computation
$\tilde{\mathbf{h}}_l = \mathbf{h}_l - \hat{\mathbf{h}}_l + \mathbf{g}_{l+1}(\tilde{\mathbf{h}}_{l+1})$
Forward updates
$\mathbf{e}_l = \tilde{\mathbf{h}}_l - \mathbf{h}_l$
$\delta_l = \mathbf{e}_l \circ \mathbf{h}'_l$
$\Delta W_l = -\eta \delta_l \mathbf{h}^T_{l-1}$
Backward updates
$\mathbf{e}^B_l = \mathbf{h}_l - \hat{\mathbf{h}}_l$
$\delta^B_l = \mathbf{e}^B_l \circ \hat{\mathbf{h}}'_l$
$\Delta B_{l+1} = -\eta \delta^B_l \mathbf{h}^T_{l+1}$

In a) Schematic of target propagation that uses perfect inverses at each layer. Synaptic weights, associated with the forward mapping are updated in order to move the forward activity vectors closer to the targets.

In b) For each layer, h_l, it was computed a reconstruction, $\hat{h}_l$, from the layer immediately above. Then, to compensate for imperfections in the auto-encoders, a reconstruction error was added, to the uncorrected target (dark blue), computed from the layer above in the backward pass.

In this sense, the **circuit learns to learn**, a phenomenon common to many proposed approximations for backprop.

Autoencoders in the NGRAD context share key principles with predictive coding models

Feature	Predictive Coding	NGRAD
Error representation	**Explicit** neurons for error	**Implicit**, via activity differences
Error transmission	**Across layers** (bottom-up)	**Within each layer** (local)
Feedback signal	Prediction from higher layer	Target activity or reconstruction
Biologically motivated	Yes	Yes
Learning mechanism	Gradient-free (often local Hebbian rules)	Local weight updates from activity differences

Both frameworks use feedback to reduce mismatch between what the system sees and what it expects/predict — and both can drive local synaptic updates from that mismatch.

NGRAD/DTP offer a theoretical bridge between machine learning and neurobiology

These algorithms **still underperform** backprop on large-scale benchmarks like ImageNet, especially in deep convolutional architectures.

It remains still unclear how the forward and backward pathways could communicate in a real biological circuit.

Despite limitations, NGRAD/DTP is a compelling proof of concept:

These algorithms send the same kind of signal in both the forward and backward directions (not neural activity vs. error gradients)

Reveal that local activity differences, not global error signals, can drive multi-layer learning.

Errors are computed locally, within each layer, based on differences in activity — a biologically plausible alternative to gradient backpropagation.

Conclusions: Backpropagation and the Brain

How the cortex learns in deep, multilayer networks remains a major open question.

Backpropagation once seemed biologically implausible and underwhelming — but advances in computing and data have made it a powerful learning algorithm.

Today, backprop is a valuable conceptual framework for understanding cortical learning — but its literal implementation in the brain is still debated.

Some mismatches with biology (e.g., spiking neurons, Dale’s law) are minor.

Others, like the computation and routing of error signals, pose deeper challenges.

NGRADs offer a promising solution:

- Avoid explicit backprop of error derivatives
- Compute learning signals locally via activity differences
- More consistent with known biological mechanisms